QL Today

Volume 2
Issue 1
May/June
1997

ISSN 1432-5454

The Magazine about QL, QDOS, Sinclair Computers, SMSQ...


```
1330 REMark ***** example ‹BT_SLEEP 'wake','Wake
up, little Suzie'› or something like that
1340 REMark ***** You need no parentheses here!
1350 BT_SLEEP 'Wake'
1360 BT_SLEEP 'Pick'
1370 BT_SLEEP 'Rjob'
1380 BT_SLEEP 'Jobs'
1390 BT_SLEEP 'channels'
1400 BT_SLEEP 'hotkeys'
1410 BT_SLEEP 'sysdef','MAKE_DIR'
1420 REMark ***** the buttons for the most
important programs which of course have to be
1430 REMark ***** HOT_LOADed or _CHPed or _RESed
1440 REMark ***** you ... the key you defined
     and give it a name of your choice, th

     HOTKEY 'û','QD-BASIC'
     HOTKEY 'ù','QD-LIB'
     HOTKEY 'à','QSP...D'
     'T','T9...'
     '','I...'
```

```
30720 DEFine PROCedure Set_CLOCK(TK_Channel%)
0725  LOCal TK_Count,TK_ShowLoop
20    LOCal TK_ClockLoop,TK_Choice
      LOCal TK_PtrVal%,TK_UpdateSize(7)
      LOCal TK_ClockPointer$(7)
      FOR A = 1 TO 7
      READ TK_ClockPointer$(A)
      DATA 10, 864000, 60, 3600, 13, 86400
      DATA 15, 1536E7, 3, 3.1536E8, 2
      DATA Ptr
```

```
OPY file$ TO filered$
PEN #3,filered$
FLEN(#3):DIM s$(8)
  c=0 TO 1-10 STEP 2
  #3\c,chk1,chk2
  chk1=$6E1A4A87 AND chk2=$6D
  WGET #3\c+8,grn:IF grn=$C800
  #3\c+8,$C801:PRINT 'Red contents
200
  END IF
  chk1=$1840000C AND chk2
  #3\c+12,grn:IF grn
  'Red border
```

QL Today

ISSN 1432-5454

German office & Publisher:
Jochen Merz Software
Im stillen Winkel 12
47169 Duisburg
Germany

Tel. +49 203 502011
Fax +49 203 502012
Box1 +49 203 502013
Box2 +49 203 502014

English office:
Q Branch
P.O. Box 7
East Sussex, United Kingdom BN41 2ND

Tel. +44 1273 386030
Fax +44 1273 381577

Editor:
Dilwyn Jones
41 Bro Emrys
Tal-Y-Bont, Bangor
Gwynedd, United Kingdom LL57 3YT

Tel. +44 1248 354023
Fax +44 1248 354023

QL *Today* is published bi-monthly, our volume begins on 15th of May. Subscriptions begin with the current issue at the time of sign up. Subscription rates are as follows:

Germany	DM 70,-
England	DM 60,- or £25
Rest of the World	DM 70,- or £30

Payment in DM (drawn on a German bank) can be made by either Cheque or Euro-Cheque, Payment in £ (drawn on an English bank) can be made by Cheque. Cheques should be made payable to Jochen Merz Software (German office) or QBranch (English office).

Credit Card holders may subscribe by either calling or sending their Credit Card number and Expiry date. Credit cards will be charged in DM (German office) or in £ (English office).

We welcome your comments, suggstions and articles. YOU make **QL *Today*** possible. We are constantly changing and adjusting to meet your needs and requirements. Articles for publication should be on a 3.5" disk (DD or HD) in ASCII, Quill or text87 format. Pictures may be in _SCR format, we can also handle GIF or TIF To enhance your article you may wish to include Saved Screen dumps. PLEASE send a hardcopy of all screens to be included. Don't forget to specify where in the text you would like the screen placed.

Article and Advertising DEADLINES are as follows:

Issue 1:	15 April	Issue 2:	15 June
Issue 3:	15 August	Issue 4:	15 October
Issue 5:	15 December	Issue 6:	15 February

Contents

Advertisers

in alphabetical order

Editorial

Dilwyn Jones

We start volume 2 of QL Today with a new look for the magazine. We've decided to use a new font for the body text in the magazine, as several people we showed a sample to said it was easier to read, more gentle on the eyes. I hope you approve of the change - let me have your comments please!

Unfortunately, I missed the Manchester and London Quanta workshops because my employer required me to work both weekends, so I missed some exciting new hardware announcements. New products from Qubbesoft P/D (Goldfire, Ethernet card), the latest 'Miracle', Pandora's Box, Jeremy Reeves' QL soundcard, the possibility of further new interfaces from TF Services for the I2C bus of Minerva Mk2, and Robin Barker's bidirectional parallel port for Aurora promise to make this an interesting period for QL users.

While the QL hardware scene is very promising, the software scene is not quite as exciting at the moment. There is the bad news that Quo Vadis Design will cease retailing software soon, but this is balanced by the good news that they will concentrate on developing new software and Qubbesoft P/D seem likely to start selling commercial software. There is actually quite a lot of software being developed at the moment, so in the months to come there could well be a lot of exciting things happening, especially once the extended colour drivers of SMSQ/E finally come out.

I joked in an article in a recent issue that someone had asked me to write a Windows style program manager front end for the pointer environment. That's taking things a bit far (apart from obvious copyright problems), but many people would like a simple to use system for launching programs and controlling the most important functions of the QL and pointer environment - one user suggested that this ought to be 'QPAC3'. Whether Tony Tebby will ever oblige us on that one remains to be seen, but I've been surprised by the number of people who picked up on this idea.

The buttons article by Geoff Wicks in the last issue (True Confessions) provoked quite a response from button enthusiasts, as you'll see. Geoff Wicks may have to eat his Just Words! That's why there are several articles about buttons etc in this issue, as it proved to be a popular subject in the end! I don't mind stirring up debate about such subjects in these pages, as long as it's constructive and ultimately beneficial to all of us as QL users. At least, after all the Button articles in this issue, we'll all be experts on the subject, and even poor Geoff Wicks may end up with a screenful of them after being converted to the cause, perhaps.

Wolfgang Uhlig is one writer in this issue on the subject of buttons, and also writes on the subject of QPAC2 boots, another very popular subject in these pages - judging by the response, people seem to be learning a lot from these occasional articles.

Please don't read too much into the cartoon on page 26 this month. It was inspired by my PeeZee (as Jochen calls them) upsetting me yet again. After all the trouble I've had with the dratted thing, something very positive came out of the trouble this month, because a Windows crash forced me to reinstall QPC, and I realised that the Windows 95 installation routines of QPC had been vastly improved over the original versions (as long as you have the up to date instructions!). The WININST utility, for example, makes it very easy to place an SMSQ/E icon on the desktop, making it easier to start QPC.

Our cover disk with the last issue seems to have been quite well received, although it meant a lot of work for all concerned. We didn't quite get it right, I managed to send Jochen a faulty ZIP file of the disk content, leading to one or two things getting left out (we ran out of space anyway), and vitally we forgot Stuart Honeyball's clock trimming routine, which we reproduce in this issue. Apologies for that. We haven't yet decided whether or not to release more cover disks, though we're thinking about it and talking to one or two people about the idea. We'd welcome your comments on the subject, and indeed your comments on any subject for the letters page in QL Today.

Congratulations to Robin Barker on his election to the post of Quanta chairman - Robin was elected at the London Quanta AGM. And our thanks to all of the Quanta officials of the last year for their hard work.

Finally, we're still on the lookout for PD software reviewers. There's such a lot of free or low cost software available for the QL that we really do need to find people who can tell the readers what's good and bad. We can provide the software if necessary, but we need volunteers who'd like to see their reviews in print. Darren Branagh from Ireland has contributed such a review for this issue, and so how about more reviews like this? No experience required, but you must be able to meet the deadlines for each issue of course (see inside front cover).

Dilwyn Jones

News, News, News

News from QUBBESOFT P/D

Qubbesoft have announced that they plan to release an Ethernet networking card for the QL. More details to follow, but Qubbesoft say that the hardware is almost ready for launch.

Qubbesoft and Di-Ren have co-operated on producing a bi-directional parallel port for the QL. Or rather, for the Aurora, since the interface will work via the ROM port and simple bidirectional working is not possible on the old QL EPROM port. If suitable drivers can be written, it would then be possible to attach parallel port hardware such as scanners and removable cartridge hard drives, as has been possible with PeeZees for some time.

The Goldfire card will now be able to support up to 256MB RAM, rather than the originally announced 64MB, using 72 pin SIMM memory cards commonly available for other computers. In addition, the Goldfire card will have a multiprocessor capability, with a second chip being used for I/O, for example, although this may need some small changes to the device drivers for boards such as the Qubbide, for example. The L-shaped Goldfire board will be roughly the same size as the Aurora. Goldfire can, of course, be used with either a standard QL (though the power demands of the SIMM boards will stretch the original power supply!) or with Aurora. The board will consist of 4 major chips, with a possible small extra board for a second processor and a GAL chip. In terms of speed, a 33MHz Goldfire should be about 4 to 5 times faster than the existing Super Gold Card, while a more expensive 40MHz could be 6 to 8 times faster than the Super Gold Card. At the time of writing, Qubbesoft had not finalised the design enough to be able to decide whether to offer the 33MHz or 40MHz processor option, as it depended on cost. The board's designer said that the 68020 in the Super Gold Card could offer about 7 MIPS (Million Intructions Per Second), while the Goldfire would offer about 30MIPS.

Ron Dunnett confirmed that although the extended colour mode drivers are not yet available for Aurora, there is now a version of SMSQ/E which does not require patching specifically for the Aurora. At the time of writing, this version (2.81) was undergoing Beta testing, looking promising.

Finally, after some reorganisation at Qubbesoft P/D, customers will now be pleased to hear that they will not need to remember two separate numbers for voice and fax to Qubbesoft. An automatic phone/fax switch now means that by the time you read this, both can be handled on the one number, 01376-347852.

QL Sound Card

Former Miracle Systems employee Jeremy Reeves has produced his first commercially available QL add-on card in the form of a low cost sound recording and playback card. This small board, which plugs into the QL EPROM slot, sticks out about 1.5 inches (nearly 4cm), and has two sockets, one for audio input from a microphone or other external source, and the other an output to low power speaker or auxiliary input of an external amplifier. The available output power is limited by the amount of power which can be drawn from the EPROM socket, about 50mA. The audio is recorded using Delta modulation, about the simplest technique possible, with 1 bit sampling at 300kHz, with the output being 6 or 8 bit (with over-sampling) quality. The card can only be used with a Gold Card or better interface, says Jeremy, and to keep costs down, there will be no through connector for the EPROM slot. On a Gold Card it will only be possible to keep about 2 minutes of audio in memory (digital audio eats up memory - professional audio equipment can use up to 10 MB of space per minute of audio). It will be possible to program the card from Superbasic and assembler, with the required software being supplied on disk. Jeremy also hopes to offer a sound filing and playback program, along with some ample pre-recorded sounds. Jeremy told me that he expected it would be possible to convert .WAV files used by Windows, giving a readily available source of audio clips. Only mono audio will be possible. There will be an audio level control on board. Jeremy is hoping that this sound card will cost about 30 pounds, though at the time of writing, the final component and manufacturing cost was not yet known, so this figure should be seen as an estimate only. The board will also be available from Q-Branch.

For further details, contact: ***Jeremy Reeves, 22 London Street, Kingswood, Bristol, England, BS15 1QZ***

News from Steve Johnson

Newly released, SJPD 71 is a disk containing a Z88 to/from PC file transfer program, which runs

on a PC (strange - supplied on a QL disk, but runs on a PC for use with the Z88-what is the world coming to!). Also on this disk are the following QL software. A program to aid the selection of numbers for the UK national lottery, a text guide to the Internet and a glossary of computer abbreviations, a pointer driven game called 23Bullets, a utility to allow the quick linking and unlinking of hard drive partitions, a file finding utility, a copier utility which can copy an entire disk, including subdirectories (and even create those directories on the backup disk), a utility to aid with character translation when transferring files between PC and QL, a remote shell program for the QL (RShell) and the latest version of the QXL Format program (v1.05) from Dave Walker. TGBak is a remote back-up program for the QL.

The 'Specials' section of Steve Johnson's catalogue also lists new clipart disks for the QL, including subjects such as farming and music on disks SJS179 and SJS180. Disk SJS181 contains v3.33 of the Ghostscript interpreter, ported to the QL by Jonathan Hudson, as described in the last issue of QL Today.

Meanwhile, Steve has been busy expanding his range of popular classic literature on floppy disk, all available for much less than you'd pay in a shop. The Classic Books section has expanded to include a number of new text based disks, mostly supplied with a simple to use on-screen viewer. D. H. Lawrence's 'Sons and Lovers' is available on CB160, while CB161 contains another translation of the Holy Qur'an. Goon Show fans will be pleased to find on CB162 a 2 disk collection of scripts from the popular British TV series. CB163 contains an unusual dictionary - a set of ASCII text files which make up a Rhyming Dictionary (don't look at me, it's not true that everyone in Wales writes poetry!!!). Other releases include The Arabian Nights, The Scarlet Pimpernel, Sinking of the Titanic and Great Sea Disasters on disks CB164-166, while tea lovers will appreciate CB167, The Book of Tea! The next two disks contain two H. G. Wells books, War In The Air and When The Sleeper Wakes. CB170 contains This Side of Paradise by Scott Fitzgerald, and CB171 is Charles Dickens' Hard Times (2 disks) and CB173 also contains a Charles Dickens book, Great Expectations. CB172 contains a Jack London novel, The Red One. CB174 contains the 14 Gilbert and Sullivan plays. CB175 contains a special version of the Probert E-text encyclopaedia reviewed in QL Today a few months ago for hard disk users only (files too big to fit on floppies). CB176 gives you A Portrait of the Artist as a Young Man, by James Joyce. And finally, CB177 contains Gulliver's Travels, by Jonathan Swift.

News from Quo Vadis Design

With effect from the 31st May 1997 Quo Vadis Design will no longer be retailing any QL Software. Quo Vadis Design will however still be active in the QL scene but as a software developer and not a supplier. Several development projects have been started and this departure will speed up the long development schedule for these programs. Alternate suppliers/publishers are being sought for the entire range of products previously supplied.

Bruce Nicholls

News from Geoff Wicks

Version 2.10 of STYLE-CHECK has now been released.

STYLE-CHECK no longer needs the Turbo Toolkit, but uses 2 short routines from the Simon N. Goodwin DIY Toolkit. Toolkit 2 should also be active. Changes have been made to the file format detection and file reading routines to eliminate minor display problems.

Three versions of the program are supplied on the STYLE-CHECK disk.

The Turbo version of 2.10 is the standard version, using an external machine code file, STYLE_cde, or 544 bytes.

The QLiberated version of 2.10 is slower than the Turbo version, but offers higher compatibility with the newer operating systems. As STYLE_CDE is built into the program, no external extensions are necessary. The program can be configured using the QJUMP configuration program. The QLiberated version of 2.10 runs about 30% faster than the QLiberated version of 2.02

For users who already have the Turbo Toolkit on their system, and who are reluctant to install another machine code extension, the OLD version 2.02 remains on the disk.

PANDORA

The North East Manchester QL Users Group have launched a custom built enclosure for the Aurora and QL based systems. Unlike conventional tower cased systems, for example, this one is especially designed with the QL user in mind, so that you don't end up with a huge box

with most of it empty inside. This enclosure is compact, robust and portable. It's made of mild steel sheet, with a durable black fine texture powder coated finish, and the steel is internally zinc coated for better electrical contact.

The case measures 175mm wide, 180mm high, and 515mm long, including the feet, etc. There's room for up to three 3.5" floppy disk drives on the front and one 5.25" drive. Hard disk drives can of course be placed in one of the floppy drive bays if required.

The case has room inside for an Aurora and a Qubide/QPlane assembly if required. The front panel can hold a serial connector, keyboard socket, mouse and even an I2C (Minerva mk 2) connector. At last! A case where I don't have to struggle round the back every time I want to plug in/unplug the keyboard, or provide an extension because the keyboard lead just isn't long enough to reach round the back of the case!

Since many QLs end up getting carted from place to place, this case has a recessed handle built into the top of the case. And the case is not just designed for an Aurora - using an optional QL adaptor plate, the larger QL motherboard can also be fitted into the case.

The case and power supply cost £89.50, while the case can be bought without a power supply (e.g. if you have a PC power supply of your own) for £68.50

The case can be supplied with or without a 200W power supply built in, and the case has a mains on-off switch and also a reset button. The rear of the case can additionally accomodate two further serial connectors, a parallel port socket, monitor and even an extra 5.25 inch drive bay if required. A range of cables and connectors are also available to assist the constructor.

And if you don't feel up to building your system into a case like this, the group will also build your system into it for you!

Further details are available from

John Southern, 40 Distaff Road, Poynton, Cheshire, England, SK11 1HN Tel: (+44) 01625-850067

or from: ***John Gilpin, at Custometal Products on (+44) 01204-305157 ext. 177, fax (01204) 303728***

News from Miracle: Ultra Gold Card

Miracle Systems have announced that development has now commenced on a successor to the highly successful SUPER GOLD CARD and are calling it the ULTRA GOLD CARD.

Not many details are available at present partly because the development is at a very early stage and may well change significantly as it proceeds. It will have the often asked for bidirectional parallel port and some other new ports in addition to what the SUPER GOLD CARD already offers. More importantly, though, it is intended to enable new programs to run over twenty times faster than they would on the SGC and, to make things go even faster still, the hardware will allow for easy connection of multiprocessor systems. It is hoped that this will encourage programmers to develop multiprocessing skills and bring us back into the lead ahead of other computer systems.

So should you go for the UGC or Qubbesoft's rival Goldfire? Well, the Goldfire is likely to be first to market, cheaper and will probably run old QL programs faster. But if you want to step into the future then the UGC will offer some really exciting opportunities.

Archivers Control Panel V3.3

The latest version of this file compression utility controller for ZIP, ARC, LHA, ZOO and LHQ is now available from PD libraries and bulletin boards. Programs like ZIP can be rather cumbersome for the less experienced user in particular to use from the basic command line, so this little front end program by Thierry Godefroy is a welcome piece of QL software.

Version 3.3 corrects a bug which was dangerous for system stability in RAM-based operating systems (SMSQ/E and UQLX, for example), and it is possible to use the Encrypt Archive option in recent versions of InfoZIP. A slight change to the zoo extension passing mechanism has also been made.

News from Jonathan Hudson

The 'giftif' program on the cover disk has been superseded by a new (and much better) tool -- "unpic". This is on my Web Page and can convert PICs in modes 2, 4 and 8 to GIF, TIF, PCX or PBM in arbitrary sizes. It is more reliable than the 'giftif' (i.e. it copes with the Progs advert, so Jochen can do these himself next time). The internal algorithms are 8bit, so upgrading to Aurora formats should be possible "just tell me where I download the documentation from".

QEYES in mode 8? Huum, having not used mode 8 since around 1986, I tend to forget/ignore its existence. As the source code is included, fixing this is left as an 'exercise for the reader'.

Apropos JMS news (p7) concerning programming examples for the new WM.RPTRT function, there is a freely distributable c68 example on my Web page 'winexplore.zip'

A version of xtc68 (c68 cross compiler, hosted

on Linux or DOS/Windows/NT/OS2 to build SMS/QDOS programs), to the most recent c68 revisions, has been uploaded to ftp.demon.co.uk/pub/qdos/.

A new version of the the 'atp' offline reader (for reading and replying to BBS messages at your leisure) that removes the random and annoying 'packet is older' message is available from my web page.

I'm working on a new version of QVM (quintessential voice mail) that will work with USR (and Rockwell based (e.g. Supra et al) voice modems. This upgrade will support all the old features (voice mail, fax on demand, data access) and "hands free" speaker phone mode for modems (like the newer USR Sportsters) that support this. Anticipated release date is April 1997.

Q Branch News

We have a limited number of 14" SVGA colour monitors adjusted for the Aurora and we will be selling these at various prices depending on condition and screen resolution from 65 to 85 pounds + carriage. QCount, the pointer driven accounts package is now released. It comes complete with the DATAdesign engine and the pointer files and will sell for 25 pounds. We have some high quality mouse mats for sale at 5 pounds We may also be making Aurora mounting kits to mount the Aurora motherboard into a standard PC tower case with the minimum of cutting. Call for details. We will also be supplying many of the programs that Quo Vadis used to supply including Flashback, Text 87, and probably the Ergon software.

Quanta News

Quanta sub-librarian Tom Mould has contacted us to ask us to print his new address for the benefit of Quanta members: Mr. T.E.Mould, 19 Bridlington Avenue, Hull, England, HU2 0DU. Tel. 01482-322917

Jochen Merz Software News

The odd SMSQ/E problem which exists on some customers' system is still not solved. Debugging code has been added to SMSQ/E V2.83, which can be activated optional. We hope that this may lead to a clue of what is going on. It seems that on most problematic systems, just activating the debugging code makes it work, which means that we're stuck so far. Don't get too desperate, we're working on it.

Future version versions of JMS-Software (we're hoping that others will follow too) will give you some nice features: you configure "your" global settings in the Menu Extension (i.e. main window colours, subwindow colours, button colours and - believe it or not - if the update sprite should flash or not) and all the programs will automatically read the settings. This means that you configure Menu only, and QD, QSpread, QMAKE, MenuConfig etc. will read the settings from there. Of course, you will still be able to set colours within the program if you want too.

We hope that other programmers will follow this idea - you can read the various values using the INFO call in Menu. Details of global settings can be found on the Config Level 2 page in the JMS-BBSs where all registered IDs can be found. Or send a QDOS-formatted disk + return postage.

I'd like to take the opportunity to apologise for some breakdowns of my BBSs. It usually happens two hours after I leave for a show abroad, i.e. Eindhoven, England etc. PBOX is quite stable, it runs for weeks and weeks as long as I'm here. Therefore, if the box does not reply for some days, then you know I'm away for a QL show.

Also, if things sometimes take a bit longer than you expect, consider: I am only a one-man band, and everything needss to be supported (i.e. fax, letters, orders, mailbox, bug fixing, shows) and not to forget the enormous amount of time that the making of QL Today requires; every inquiry/letter/problem will be dealt with, but it may take some time.

■

2 Shows 1 Weekend

Stuart Honeyball

There have been quite a few meetings recently and the weekend of 19/20 May was no exception. In fact there were 2 meetings; fortunately both in England so traders could easily get to them both.

I met up with Jochen Merz and Bernd Reinhardt in London on Friday afternoon and we travelled up to Stafford where we arrived in the early evening. Some of the others, namely Roy Wood, Tony Firshman and Bill Richardson had already checked into the hotel when we got there. After a pint in the bar we ventured out to sample the wonders of Stafford and came across an Italian restaurant. We were told that it was full despite there being only 1 table occupied so we arranged to come back 1/2 an hour later. We passed the 1/2 hour in a huge pub that had formerly been a cinema. There was a good selection of real ale (which didn't go down too

well with the 2 continentals!) and no musak so scored 10/10. On arriving back at the restaurant there was still only 1 table occupied but this no longer counted as being full so we were able to have dinner. The house red we ordered turned out to be vinaigrette and had to be sent back. None of us were impressed by this establishment.

On the Saturday we arrived at a cow-shed just outside Stafford and set up shop in a special area that John Taylor of Quanta had negotiated with the organiser to be solely for the QL. This area was occupied by Jochen Merz Software, TF Services, Q-Branch, Quanta, Qubbesoft and W. N. Richardson & Co.. Simon Goodwin turned up and put some effort into trying to track down a programming problem using Jochen's system - the FPU extension does not yet want to run properly on the TT. Anyway, the rest of the hall was taken up by a radio rally. These are good places for getting bargains if you're looking for monitors, connectors, batteries or even radio gear.

On Saturday evening we travelled down to London to the Victory Services Club where there was overnight accommodation as well as the room for the Quanta workshop itself. On the Sunday the room soon became full with standholders and users and trade was generally good. The London sub-group was very much in evidence with a few people from further afield. There were more traders than at Stafford amongst whom were Custometal Products showing off their PANDORA. This is a well styled case for housing a QL/Aurora system being a better alternative to a PC case. The advantages over the PC case are that it looks the part, is more compact, is better designed having air intakes and a carrying handle and is easier to put together. Custometal can be contacted by 'phone on +44-1204-305157 or fax on +44-1204-303728.

In the afternoon the Quanta AGM started 1 hour later than advertised. To me the Quanta committee resembles a dog chasing its own tail; it expends a lot of effort but doesn't really achieve much! The meeting got off to a bad start with Phil Jones (magazine editor) questioning the accounts. There was a lack of clarity which led to some members voting against accepting them. The options on the voting forms for committee members sent out are unique in my experience. For each person standing for election you can vote 3 ways: For, Against or Abstain. There is of course a 4th option and that is not to vote at all so you can abstain either by voting "Abstain" or leaving all 3 boxes blank. We also learnt that if you voted "Against" or "Abstain" then it would be treated as if you abstained. Why, therefore, have these extra options I asked John Mason (secretary) to which came the one word reply: "Precedent". Plenty more tail-chasing occurred throughout the afternoon until the time the hall had been booked for ran out with several issues unresolved. We did however get an untertaking from the committee that they would be more open and less secretive in their workings in the future.

All in all it was a busy but worthwhile weekend.

■

Buttons - more Comments

Dilwyn Jones

Geoff Wicks' article about the Button frame generated a lot of messages and correspondence, and some of the articles in this issue reflect that response. Two points were raised in a telephone conversation to me by a gentleman who asked that I print these details in the magazine, but unfortunately I forgot to write the man's name down and promptly forgot it (sorry). Here are two useful tips for users who like to use the button frame.

1) Even if a program has no sleep or ZZzz icon, the Button_sleep thing in QPAC2 can put most programs to sleep. Simply call up the EXEC menu in QPAC2 and choose BUTTON_SLEEP. If you can remember hotkey definitions quite well, it can be useful to define a hotkey to activate button_sleep as required. ERT HOT_WAKE(CHR$(233),'BUTTON_SLEEP') So every time you press CTRL-ALT F1 it zaps the current program into a button in the button frame. To reactivate the program, simply move the pointer over the program's button and press ENTER or the right mouse button.

2) Remember that QPAC2 can be configured to make the button frame be organised as either rows or columns. Now that larger screen displays are commonplace (VGA and SVGA on QXL, QPC or Aurora systems, for example), it can be advantageous to organise the button frame vertically as long as your software can be moved to the right of the screen. This means that the little used area to the right of the screen can be used by programs, while the button frame uses the left of the screen.

■

Clock Trimmer (Part 2)

Stuart Honeyball

Many apologies for the missing listing in issue 6 of volume 1. The interest was big, so that we cannot wait until the next cover disk comes with QL Today. Therefore, the listing:

```
1000 JOB_NAME "ClockAdjuster_bas"
1010 :
1020 REMark This program helps the RTC
1030 REMark to maintain its time
1040 REMark more accurately.
1050 :
1060 TimeFile$="Win1_TimeFile"
1070 :
1080 TK2_EXT
1090 NotFound=-7
1100 :
1110 DEFine PROCedure CheckTimeFile
1120   REMark Call this once a week
1130   LOCal TFCh%,Response,SecondsOut
1140   LOCal StartTime,AdjSoFar,CorFactor
1150   LOCal StartTimeHigh,StartTimeLow
1160   LOCal AdjRequired,TimeElapsed
1170   TFCh%=FOP_IN(TimeFile$)
1180   IF TFCh%=NotFound
1190     REMark File not created yet
1200     PRINT "Confirm the clock time is correct"
1210     PRINT "before the Time File is created"
1220     PRINT "Is the clock correct? y/n"
1230     REPeat WaitCorrect
1240       Response$=INKEY$(-1)
1250       IF Response$=="y"
1260         REMark Clock is correct
1270         TFCh%=FOP_NEW(TimeFile$)
1280         IF TFCh%<0
1290           PRINT "Can't create"!TimeFile$
1300           RETurn :REMark leave procedure
1310         ELSE
1320           StartTime=DATE
1330           StartTimeHigh=StartTime DIV 65536
1340           StartTimeLow=StartTime MOD 65536
1350           AdjSoFar=0
1360           CorFactor=0
1370           PRINT #TFCh%,StartTimeHigh\StartTimeLow\AdjSoFar\CorFactor
1380         END IF
1390         CLOSE #TFCh%
1400         PRINT "New"!TimeFile$!"created"
1410         RETurn :REMark leave procedure
1420       END IF
1430       IF Response$=="n"
1440         PRINT "Use SDATE or ADATE to correct it"
1450         PRINT "then start again."
1460         RETurn :REMark leave procdure
1470       END IF
1480     END REPeat WaitCorrect
```

```
1490   ELSE
1500     IF TFCh%<0
1510       PRINT "Can't open"!TimeFile$
1520       PRINT "So leaving procedure."
1530       RETurn
1540     ELSE
1550       REMark Update CorrectionFactor
1560       INPUT #TFCh%,StartTimeHigh,StartTimeLow,AdjSoFar,CorFactor
1570       StartTime=StartTimeHigh*65536+StartTimeLow
1580       CLOSE #TFCh%
1590       TFCh%=FOP_OVER(TimeFile$)
1600       PRINT "How many seconds adjustment required?"
1610       PRINT "(Use + if slow, - if fast)"
1620       INPUT AdjRequired
1630       ADATE AdjRequired
1640       TimeElapsed=DATE-StartTime
1650       AdjSoFar=AdjSoFar+AdjRequired
1660       CorFactor=AdjSoFar/TimeElapsed
1670       PRINT #TFCh%,StartTimeHigh\StartTimeLow\AdjSoFar\CorFactor
1680       CLOSE #TFCh%
1690       PRINT "Update done."
1700     END IF
1710   END IF
1720 END DEFine CheckTimeFile
1730 :
1740 DEFine PROCedure CorrectClock
1750   REMark Call this every boot up
1760   LOCal TFCh%,Response,SecondsOut
1770   LOCal StartTime,AdjSoFar,CorFactor
1780   LOCal StartTimeHigh,StartTimeLow
1790   LOCal AdjRequired,TimeElapsed
1800   TFCh%=FOP_IN(TimeFile$)
1810   IF TFCh%<0
1820     PRINT "Can't open"!TimeFile$
1830     PRINT "so clock not corrected."
1840     RETurn :REMark Leave procedure
1850   ELSE
1860     INPUT #TFCh%,StartTimeHigh,StartTimeLow,AdjSoFar,CorFactor
1870     StartTime=StartTimeHigh*65536+StartTimeLow
1880     CLOSE #TFCh%
1890     TFCh%=FOP_OVER(TimeFile$)
1900     TimeElapsed=DATE-StartTime
1910     AdjRequired=TimeElapsed*CorFactor-AdjSoFar
1920     ADATE AdjRequired
1930     AdjSoFar=AdjSoFar+AdjRequired
1940     PRINT #TFCh%,StartTimeHigh\StartTimeLow\AdjSoFar\CorFactor
1950     CLOSE #TFCh%
1960     PRINT "Clock corrected"
1970   END IF
1980 END DEFine CorrectClock
```

■

Sorting Routines - Part 2

Dilwyn Jones

This is part 2 of the Sorting Routines article. It refers to listings on the Cover Disk which came with Issue 6 of Volume 1.

1. EXCHANGE SORT

The exchange sort is just about the simplest of all sort routines. Unfortunately, it is not the most efficient, although for small amounts of data its brevity may make up for its lack of speed. It is a routine called EXCHANGE_SORT in the sort_TEST_bas program, also a routine called sort_EXCHANGE_SORT_bas on the disk (use the LOAD command to load it). Similar naming conventions are used throughout this article.

The exchange sort is roughly comparable to the bubble sorts in timings, but has the advantage that its performance does not vary greatly depending on whether or not the data is already in a well sorted order, so its performance is at least predictable. Note that its sort timings (like most sorting routines) are not linear – if you double the amount of data to be sorted, you will find that the amount of time taken more than doubles and will deteriorate badly if used for very large arrays.

The first pass through the array finds the lowest entry and moves it to the top of the array. From then on, further passes find the lowest entry each time and move it up to the top of the section being sorted, and this goes on until all of the array has been sorted.

2. BUBBLE SORT

Just about everyone has heard of the Bubble Sort algorithm. It is short, simple to program, almost as simple to understand, and is fast enough for small amounts of data.

Depending on the existing sort order of the data, a bubble sort can be considerably more efficient than an exchange sort. During each pass of a bubble sort, one element is carried to the head of the array, being swapped with elements which as a result of the swap move closer to their sorted position in the list. Thus the elements gradually "bubble" into their sorted area.

The basic bubble sort is shown in the program called BUBBLE_SORT1. To load this, use LOAD FLP1_sort_BUBBLE_SORT1_bas.

That routine does the job, but it can be improved on a little. The bubble sort becomes less and less efficient as the amount of data to be sorted increases. If you double the amount of data, the bubble sort takes more than twice the amount of time to sort it.

If nothing is swapped during a pass of the inner loop, it means nothing is out of order and the sorting can be ended earlier. Also, during the inner loop we may find that the array element referred to by the outer loop is already in its correct position. This can be detected while the inner loop is being executed by keeping a note of the number of the last record moved down. So we can alter the routine as in BUBBLE_SORT2 by using the variable "last_swap%" to hold this value.

One further way of improving the bubble sort is to bear in mind that the routine is not too bad for small amounts of data. So if we split the data to be sorted into smaller sections, sort those, then merge the resultant sub-arrays we can improve on the performance a little bit for larger amounts of data. Note that for small amounts of data, the processing time may actually be a little slower, but this is worth it for the improvement in sorting times for larger arrays. This is shown in BUBBLE_SORT3. We could call this algorithm a 'double bubble' if you like, since it performs a separate bubble sort on the first and second halves of the array to be sorted, before the two halves are later merged into order. I am grateful to Joe Haftke for suggesting this technique to me. We will find that some of the more complex sorting routines use fairly similar principles to split data up into smaller sections for sub-sorting. Note that BUBBLE_SORT3 calls BUBBLE_SORT2 to sort the array halves, so if you intend to use BUBBLE_SORT3, you should also have loaded BUBBLE_SORT2 (merge them).

Because BUBBLE_SORT2 has better performance where the data is already in a fairly well sorted order, it may prove adequate in applications where a small amount of data needs to be added to an already sorted list, e.g. adding one or two new customer names to a business address list.

3. INSERTION SORT

One further modification we can make to the bubble sort to improve performance is to alter the way in which the values bubble into the right place in the array. If two entries are found not to

be in the correct order, we can move the second one back past all those elements which should come after it in the list. temp$ holds the value of the first out of order entry. So rather than bubble through the list one entry at a time, out of order entries are sifted as far as possible with each move, which can greatly increase performance compared with the standard bubble sort routine. This routine is also sometimes called a Sift Sort, the basic idea being that elements are scanned once, then sifted back through the array until their correct insertion point is found. Despite normally being called an Insertion sort, the routine is basically a type of Bubble sort.

Although I have not attempted to write such a routine, you may find it an interesting exercise to try to implement a Bubble sort variation which takes advantage of the fact that sorts like this are fairly good at sorting small amounts of data, or greater amounts of data if already well sorted. So you could try splitting the array up into sections (some trial and error needed to determine best actual sizes of these sections) then gradually merge the sections by successively sorting adjacent sections to produce better ordered data before the routine has to tackle large amounts of data. In fact, this approach is not far removed from the methods used by some of the more advanced sorting routines, but it can still be an interesting way to learn about sorts by tinkering with Bubble sorts, trying to squeeze the last possible bit of performance out of them.

I have gone into some detail on the bubble sort, since it is easy to understand, and the ideas behind these variations I have shown will help you to understand some of the principles behind the more complex sorting routines.

4. SHELL SORT

The Shell sort (also known as Shell-Metzner) is a significant improvement over the types of sorting routines discussed so far. While not as fast as the more advanced Quicksort and Pigeon Sort, Shell is easier to program and understand, while remaining fast enough for most applications. This listing is also quite short given the efficiency of the routine.

The Shell sort divides the array into smaller sections, swapping out of order entries as required, and progressively changes the sizes of the sections sorted then finally does one sort on the whole array, which by that time is quite well in order and so fewer time consuming exchanges are required.

The Shell sort is not particularly fast compared to the earlier sorting routines when used on fairly small amounts of data, but the relative speed gets better and better as the array size increases.

The routine is called sort_SHELL_SORT1_bas on the disk.

5. QUICKSORT

The Quicksort routines make use of a programming technique called recursion, where the routines break down the array to be sorted into sections and call themselves to sub-sort these sections. There was an article about recursion in QL Today volume 1 issue 3. Two variants of this type of sorting routine are presented here. The second is faster, but has a special requirement, that one extra end-of-array marker is required which has to be the greatest value entry in the entire array (e.g. if you are handling text only, this last entry could be something like "zzzzzzzzzz"). QUICKSORT1 is based on a routine written by Alan Turnbull, while QUICKSORT2 is based on a variation written by Marcus Jefferey.

The routine begins by partitioning (or dividing) the array into two parts and selecting a midpoint value. After one pass, all items less than this value are placed before it, while the greater values go after it. The routine calls itself in turn for these parts until sorting is completed. Note how the two calls to itself are used to subdivide the area further until the smallest size is reached, and to sort the respective halves of the array to be sorted.

If the data presented for sorting is in a fairly random order, the routines work quite well. But use the test routine to see what happens when either fully sorted data, or completely reverse order data (descending values).

While the two Quicksort routines are very fast and reasonably compact, they do have one disadvantage when sorting very large amounts of data. Because of their recursive nature, they tend to use up a lot of stack space to store return details from each call. When used in compiled SuperBASIC this may prove to be a problem, so you may need to alter the amount of stack available to the compiled program to prevent out of memory or out of stack space errors.

6. PIGEON SORT

This is the fastest sorting routine of which I am currently aware. The listing was originally sent to me by QL Today contributor Mark Knight, who asked that I include the following line of credit:

Pigeonsort was originally written by Paul Birch in 1988. Pigeonsort made history as the first general purpose sort that sorts in O(n) time (of order of n), a feat previously "proved" to be impossible by mathematicians. The original program was written in Fortran-77 for a mainframe computer, and later converted to QL SuperBASIC by Brian Mewse

This listing has not been built into the test and timings program, for the simple reason that I don't as yet fully understand how it works! So I have simply placed it on the disk in the same form as supplied by Mark, for fear of causing errors in the listing if I attempted to rewrite it without fully understanding how it works.

It is designed for sorting numeric arrays only, and also requires an index array (a second list, which specifies which order the elements of the first array should be indexed). Also, please note that element 0 is not used during the sort - only elements 1 to NMax are used. Load the file flp1_sort_PIGEONSORT_bas to study the program. A look at lines 200 to 340 will show you how to call the routine. The sort routine itself is completely contained within the procedure called PigeonSort, in lines 10000 to 11470. It should be called with two numeric arrays and the index number of the highest entry in the array. In this listing, the array of numbers to be sorted is called DataItems(), and this is referenced by an index array called DataIndex(), with the highest element number contained in NMax.

The routine appears to work by setting up a list of slots or pigeonholes (hence the name) into which the various data values are sorted and grouped - compare this with manual sorting of mail, for example, where letters are placed in one of a series of boxes depending on the name or destination.

A separate timing routine is built into the Pigeonsort program, so you can get some idea of its performance. The listing is quite long, and in use, the routine requires a lot of array space. It is not recursive, so the potential problems with using the Quicksort routines in compiled BASIC programs, for example, should not arise. However, it should be noted that as it stands, the Pigeonsort routine cannot sort string data, since it is geared up to sorting by analysing the spread of numeric values. If anyone understands the routine enough to adapt it for string sorting, QL Today will consider publishing an extended version.

Mark Knight has asked me to point out that "doubling the number of items will only double the sort time if the first number is greater than about 30". As yet, I do not understand the significance of this, but I'm sure someone will enlighten me in due course!

In terms of relative timings, sort routines can be loosely categorised as follows. The simple bubblesort tends to slow down in proportion to the square of the number of items to be sorted, or of the order of n*n, or O(n^2), while the quicksort is much better, being of the order of n*LOG(n) or O(n LOG n). Many routines exhibit special behaviours if the amount of data to be sorted is small, or if the data is already in a well ordered state, or if the data is in almost exactly opposite state to that desired after sorting.

The test and timing routines given allow you to state the size of the array to be sorted, and also enable you to test performance with random data, or data already sorted into ascending or descending order, which should be quite useful when trying to determine which routine is best suited to your needs. Broadly speaking, here are some very general guidelines to help you choose which routine is best suited for your needs.

(i) Simplicity, brevity, ease of modification and modest amounts of data - try the bubble or exchange sorts.

(ii) General use where speed is important, but the routine needs to be fairly consistent in performance and fairly easy to understand and modify - try the Shell sort.

(iii) Speed is important, recursion is not a problem, fairly large amounts of data to be sorted - use the Quicksort routines.

(iv) Numeric data, with speed being vital, and the size of the listing and the amount of array space required being no major problem - use the PigeonSort.

SAMPLE TIMINGS

Although I don't want to spoil your fun too much, I have included below a table of sample timings for all the sort routines, using the 'random' data option. This was carried out using uncompiled SuperBASIC on a version JM QL with a Super Gold Card. Obviously better results

could be obtained with either compiled SuperBASIC or SBASIC on SMSQ/E. Hint: when compiling these routines, use the DEF_INTEGER or IMPLICIT% directives on your compiler to make as much use of faster integer loop counts and variables as possible. The timings given are in seconds, of course.

	NUMBER OF DATA ITEMS SORTED				
SORT NAME	50	100	150	200	250
Exchange	1	5	11	21	32
Bubble 1	3	10	22	38	61
Bubble 2	2	9	21	38	61
Bubble 3	2	6	14	25	39
Insertion	1	4	9	16	25
Shell	2	2	5	6	8
Quicksort 1	1	2	4	6	7
Quicksort 2	1	1	2	3	4
Pigeon	1	1	2	2	3

INSERTING NEW ITEMS INTO SORTED ARRAYS

There is one other aspect of sorting I would like to touch on. I've mentioned that some sorting routines do quite well if the data is largely in order already, so that few or no swaps are actually required. Databases, for example, are often held on disk in sorted order (e.g. a list of names and addresses may well be held sorted by surnames). Occasionally, new names need to be added to the database. It seems wasteful to have to make a complete sort which may take a minute or more after adding just one or two names, so here is a useful little technique for inserting new entries into a sorted array, maintaining the sort order as it goes along.

The basic principle is to look at the array elements until we find the entry which should go after the new entry. If there are some entries, but none to go after the new entry, that means that the new entry has to be added at the end of the array. If the first entry in the database is greater than the new entry, that means everything has to be shuffled up one place to make room, and the new entry placed at the start of the array.

We will look at two techniques for achieving this. Firstly, the linear search from the start of the array until we find the required point. This is adequate for small amounts of data, but becomes quite slow for larger amounts of data, since each and every data item has to be searched for each new entry. This program is called sorts_ADD_LINEAR_bas. The program sets up a loop to count through the array entries (called 'index'). If the given entry is found to be higher than the value of temp$, the loop called 'shift' moves everything from here up one place to make room for the new entry to be added. It then exits from the 'index' loop. Note that if no insertion point was found, it means that the new entry was higher in value than all the existing value in the array, so this special case is handled in the loop epilogue between NEXT index and END FOR index.

How would we go about solving this problem in real life? Let us imagine we are looking for a word in a dictionary. We might open the dictionary about half way through and realised that we opened the book too far, the word we require comes before that. So we then try a quarter of the way through the book. This time, we are not far enough into the book to find the word we wanted, so then we go half way between the two points and so on until we find the word. Successively halving the distance between search points like this is called binary chopping, and where large amounts of data need to be searched, this can be much more efficient than a simple linear search. The example program is called sorts_ADD_BINARY_bas on the disk. Load either of these programs into basic and run them to see how they work. They set up an array (called array$) which can contain up to 10 entries.

Each entry can be up to 10 characters long. The new entries are typed into the variable temp$, which is then added to array$ by the ADD procedure. A running total of the number of words entered is held in the variable number%. The idea is that you enter up to 10 words, in random order, and the program maintains a sorted list as it goes along. Line 200 contains a PRINT statement which prints out the entire array on one line so you can see for yourself that this does really occur.

The binary search determines the start and end points of the data in the array. The variables lo% and hi% hold this information - note the use of integers, since arrays cannot hold more than about 32,768 entries, and integer variables can hold this size of number, it makes sense to use integer variables where possible if you are using a version of basic or compiled basic which can take advantage of speed improvements through use of integers, and sorting is a

very time consuming affair after all. Then a mid point is selected, roughly halfway between these two values, the variable mid%. The array entry at this point is checked against the variable temp$, and depending on whether larger or smaller, the start or end points of the search are adjusted accordingly (the routine changes the value of lo% or hi%). If this results in a crossover of the values of lo% and hi%, it means that there was no entry already in the array which matched that in temp$, so we can now insert the entry at the point indicated by mid%, UNLESS the value of mid% has gone below that of lo% in which case it is corrected in line 380. Lines 410 to 430 move entries from the insertion point to the end of the array up one location to make room for the new data to be inserted at the correct point to maintain the sorted order. But if we had not yet found the insertion point, we go back round the loop and set a new value for mid% roughly halfway between lo% and hi%, and try again.

That concludes this article on sorting techniques. I hope that you have found it useful.

■

15 Common Programming Mistakes

Rich Mellor

I have been writing, testing and fixing computer programs for several years. What follows is some general advice based upon my experiences working with other people's code and should hopefully assist anyone who is hoping to write a successful program. It has been written with the intention of being applicable to as many different types of computing environments as possible, we do after all live in a world which is not totally dominated by Pentium processors! ***[Don't mention that word! - editor]***

We all make mistakes when writing a computer program, and it is very rare to find a program which does not contain bugs (even if that program has been on the market for years). However, some of the mistakes are so fundamental that a programmer should be aware of them before setting out on writing a program:

1. Assuming the user will read the manual

This is one of the largest areas of conflict between programmers and users - having spent hours writing the manual, the programmer wants the user to read it (what, you mean to say that you didn't spend very long on the manual?). Manuals tend to be fairly cumbersome and not very exciting reading (this is no comment on the authors!) and on buying a program, users just want to load it and see what they can do with it.

When testing programs, I only look at the manual as a last resort (normally only to see how good it is!) - the reason for this is that quite often, the programmer assumes that certain things within a program can only be performed one way (the way set out in the manual). Normally there are ways around this and quite often exploration of the program in this way can show up various bugs which would not have been revealed had the user followed the manual (this is after all the programmer's way of doing things (but not always!)).

Having said this, it is still imperative that as much thought is given to the manual as to the program itself - get someone else to read the manual (as well as trying the program) before it is released onto the market and if necessary get someone else to write the manual for you. The manual is the user's only hope of getting the most from your program and without a good manual, the programmer or publishing house will only face a string of annoying little queries.

2. Using the wrong method of asking the user to enter a number

You should have noticed that if you use a command such as:

```
INPUT value
```

from BASIC, if a valid number is not entered by the user (such as 1x0), an error will occur. It is not up to the user to know what the program expects, but for the programmer to ensure that the program can cope with whatever the user enters. The solution is to use a string, such as:

```
INPUT value$
```

and then for the program to test the characters within the string to see if they form a valid number. The value should also be tested to ensure that it is within the ranges allowed by the program.

3. Forgetting that letters in UPPER CASE have different character codes from letters in lower case

It is all too easy when using a keyboard to press CAPS LOCK by mistake and enter letters in upper case - the programmer needs to ensure that the program can understand both

upper and lower case (this is often quite a problem in adventure games).

The problem is exacerbated on the PC where CAPS LOCK forces the computer to read the characters above the number keys (0-9) rather than the numbers themselves!!

4. Failure to properly document the program

I have been sent various programs to update/test and when I have looked at the source code, it can quite often be impossible to tell what the programmer was intending to do with various parts of the code. Quite often there are whole sections of the source code which are never called or which just do not have any effect.

The problem seems to be that programmers are reluctant to keep remark statements in their source code (these can always be deleted in the release version or prior to compilation). As a result new code is added (or even duplicated) and when any bugs are reported, the programmer may not know which lines have been altered since the last release of the program which may help to track down the cause.

A well-structured program makes it much easier to find and fix problems - after all, the programmer must always bear in mind that he may have to return to a program after a period of months (or even years) and try to remember what each part of the program does. It may not even be the original programmer who is asked to maintain the program.

5. Testing of the program

Quite common errors in a program can be easily overlooked by the programmer because the program has only been tested by one person or on one system. It is always useful to get the program tested (via friends or otherwise) on as many different set-ups (memory, processor, display mode, operating system and other software) as possible. Programmers should never assume that they will have switched the machine off before loading a program.

Other difficulties arise due to programmers failing to test the whole program again after alterations have been made. Even minor changes, whether they are bug fixes or enhancements can affect the operation of the program in other areas - consider the case of a bug due to the wrong variable being used - if the value of that variable was altered as a result of the bug, another part of the program may have relied on this new value of the variable to work properly. Therefore, by fixing the bug, you have created a problem elsewhere because the variable never reaches the correct value!

It can also help to test the program fully at each stage of development - the earlier a bug is spotted the better.

6. Presumptions about the type of input device to be used

If a program will support more than one type of input device (for example a mouse and the keyboard), try them both for all commands - quite often I have come across programs where a command will work when selected by the mouse but not by the keyboard. Although use of standard menuing systems may reduce the chances of this happening, don't forget that most menu systems allow the programmer to dictate how keypresses are to be interpreted and which keypress is available to call each menu item.

7. Failure to keep track of what each variable is used for

Again, this comes down to good housekeeping - quite often bugs in a program can remain hidden for quite a while where the wrong variable has been used (or even misspelt).

8. Top-down design of programs

Many programs (especially games software) seem to be designed starting with the graphics design, normally in conjunction with the idea behind the game and with the programming forming a poor third place. This can lead to a lot of unplayable games as the programmer struggles to write code which can meet the needs of the program design. The programming needs to be given as much thought as the other two areas, which can in some circumstances mean that the programmer and program designer (if not the same person) must closely liaise from the very start - if the limits of the programmer are known at the outset this can save a lot of problems later on.

9. The need for speed

A lot of program source codes become incomprehensible as the programmer strives to get the maximum speed out of a program - shorter variable and procedure names do not affect the speed greatly (even in interpreted code) and only reduce readability. If you are insistent that the shortest names possible should be used, why not use full names in the source code (and full remark lines) and then run the program through a filter program which will convert the variable names to their shorter

equivalents and remove remark lines before compilation/distribution?

For similar reasons, it is always worth ensuring that the program is completed and works 100% before trying to optimise the program for speed - after all, it is only really the most frequently used and time consuming parts of a program that are worth optimising for speed.

10. Re-inventing the wheel

A lot of programmers suggest that they either do not know what facilities are available to their programs from the operating system or seem to take delight in re-writing whole sections of code. If a facility needed within the program already exists from another widely available source, consider using this other source as there is less code for the programmer to write and therefore less chance of bugs creeping in. This is especially helpful with printer drivers where there are a whole range of printers available, each needing its own specific printer driver. Most users will have adapted a printer driver already for their own printer and therefore why not make provision to allow the user to add the printer driver they already use? Even if the you don't want to incorporate someone else's code (for example if you wanted to avoid paying the royalties), write your own code by all means, but why not give the user the option of replacing your printer driver with one they may already have from some other means.

11. Trying to have a monopoly on the system

More and more users are experiencing the joys of proper multitasking (or even multi-processing) whereby several programs are being run side by side sharing the same facilities. It is always a shame when one of those programs assumes that they will be the only ones accessing the computer's facilities and therefore lock the other programs out - for example many computers will allow a program to open read-only files which do not prevent other programs from accessing that same file at the same time.

12. Trying to use undocumented system calls

The main problem in using anything which is not officially documented is that it is liable to change (or not be supported by emulators)! You should avoid using undocumented code as you can never be certain (a) what that code will do in all circumstances; and (b) whether that code will always be available - This is a dangerous way of cutting corners and should never be used - all that you are doing is ensuring that your software is not future-proof.

13. Use of other people's code

In a similar vein, many programmers seem to use other people's code (for example operating system calls) without fully understanding the code - if you cannot be certain how code will react in all circumstances, you need to be very careful in using it - again, this is where a lot of testing of a program is imperative.

14. Making assumptions about the system on which the program will run

Some programmers seem to make assumptions about the set-up of the system on which a program will run (for example the hardware on which the program will be used). As machines are developed, so are emulators which allow people to run programs written for one operating system on other hardware platforms. If you can write code which is hardware independent, you will increase the potential market for your program. To some extent, even operating system programs can be written (by using C or PASCAL for example) - unfortunately these programs tend to be limited as not all implementations of a language are the same and each operating system may need separate libraries writing - this does however reduce the amount of work needed to convert a program to another operating system if you wish to do so.

Unfortunately, hardware and operating system independent programs tend to be slower than programs specifically written for one platform - it is a trade off between the available market and speed. Normally it is better to concentrate on making programs hardware independent.

15. Writing too much code too fast

A lot of programs appear cumbersome, not written in clearly defined sections - if you can test each section of a program separately, this will assist in tracing and fixing errors - it also useful to get a basic working program before you turn your attention to the detail (such as the layout of the menus).

Finally, may I make a plea on behalf of the users. Don't ever make the fatal mistake of believing a program to be finished and completely error free - even the oldest programs on the market still show signs of bugs every now and then or incompatabilities as the platform they run on changes. If you want people to use other software written by you in the future, support your users!

■

Professional & Graphical Software

ProWesS

ProWesS is a new user environment for the QL. ProWesS is short for "PROGS Window Manager", but it is much more than that. Apart from a new window manager, it contains all the system extensions from PROGS, and is essential if you want to run programs which need these extensions.

The ProWesS reader is a major part of the package. It is a hypertext document browser. This means that text files which include formatting commands (including pictures) and possibly links to other files can be displayed and read in this program. This is used in ProWesS to read (and possibly print) the manuals, and display the help files. The hypertext documents which are used by the ProWesS reader are in HTML format, the format which is popular on Internet to display World Wide Web pages.

Another important aspect of ProWesS is the possibility to allow programs to automatically install themselves on your system, and to be able to run them without resetting the system. This means that, when you get a new program, all you have to do is insert the disk and indicate "start the program in flp1_", a menu option in the "utilities" button. To install a program, you indicate "install software", and the software can be added to your system. This way, you don't need to know how to write a boot file to use the multi-tasking capabilities of your computer.

ProWesS includes many programming libraries. These include syslib, an interface to the operating system, PROforma, a vector graphics system, allowing rendering both on screen and on paper (via a printer driver). The DATAdesign engine is also part of ProWesS. It is a relational database system with a bonus, as you don't even need a key field. You get a powerful record at a time data manipulation extension to the language you already use. Of course it also includes ProWesS itself, the new resolution independent window manager.

PFlist

Easy to use program to create listings on any printer (especially inkjet and laser). This ProWesS application allows you to indicate the files which have to be printed. Each column contains a footer which can include the filename and filedate. The listings always allow perforation. PFlist can create your listings in two columns and in landscape (or both).

fsearch

File search utility with many useful options, like the choice to search only files with a certain extension, and whether or not the directory tree has to be scanned. All occurences of the searchstring will be displayed with line number or offset. You can also use special matching features, like case dependent, matching a space with a stretch of whitespace, and searching for a word dilimited string.

font-utils

manage your font collection. You can preview fonts on screen, see what characters exist in a font and convert Adobe Type 1 and similar fonts for use in ProWesS.

DATAdesign

Never before has it been so easy to create, fill in and maintain your personal databases. To start a new file, just type the names of the fields. To add or delete a field, no problem, just do it. To change the name of a field, just indicate it.

What's more you can choose to look at only those fields you want, and in any order you specify. And you can select which records you want to view, and which not.

DATAdesign allows you to have some hidden comments for each record, have a general look at the file (in tabulated form) or to transfer a record into the scrap of hotkey buffer, so you can easily import a record in your favorite text processor or editor !

Security is a strong point for DATAdesign. Usually files will be memory based, for maximum speed. Files can also be disk based, making sure all changes are immediatly stored on disk, so even in the event of power failure, you can at most loose the changes to one record !

Naturally, DATAdesign is good at sorting and searching. And if you were using another database, you can convert Archive or Flashback files to DATAdesign.

The new v4 of DATAdesign makes the program even easier to use than before. You can now also have QD-style icons on your screen to make the program even easier to operate.

LINEdesign

This is the program which brought QL computing into the nineties. Finally you get access to a modern technology, "vector graphics." This means that your page will be stored in a mathematical form, and not as a collection of pixels.

With LINEdesign, you can create artistic drawings, technical drawings, process bitmaps (even scale and rotate them!), and any kind of vector drawings. You can draw lines, curves, circles, ellipses, pies, squares, rectangles, rectangles with rounded corners, and any combination of these to create the most fabulous drawings ever seen. Because LINEdesign is a vector drawing program, any part of the picture can be moved, scaled, rotated, slanted without any loss of precision or resolution. In LINEdesign, pictures are device independant, meaning that the printout will be the same on any printer (e.g. same size and position).

Also LINEdesign is good at handling text. You can easily put titles and full paragraphs on the page. All the fonts can be displayed at any size, rotation, etc. All the fonts which are available to ProWesS can be used in LINEdesign.

LINEdesign is a drawing program, but it can also be used by people who are not good at drawing. LINEdesign is a great program for making leaflets, posters, and any kind of printed work. Lots of clipart and extra fonts are available from public domain libraries and BBS's. You can even import Adobe Illustrator files.

All our software has electronic manuals, which can be read and printed in the ProWesS reader. However, we can also supply printed copies of the documentation (or even your own HTML files). The costs are BEF 3 per page, plus postage costs. Contact us for more details.
ProWesS does not include the programming documentation. This is available via bulletin board and public domain software suppliers. The programming documentation is readable in the ProWesS reader, and partly in DATAdesign (the demo version is be included). We can supply the programming docs for BEF 100 (HD disks only!) If ordered with something else, you don't have to pay extra postage.

ProWesS - BEF 2400 DATAdesign - BEF 1200 LINEdesign - BEF 1200 fontpack - BEF 3000 fontutils - BEF 1200 PFlist - BEF 600 fsearch - BEF 600

Payment terms :

You have to run ProWesS to make LINEdesign, DATAdesign, fsearch, fontutils and PFlist work (even though DATAdesign uses wman).

All our software is normally supplied on high density (HD) disks. However they can be obtained on double density (DD) disks at an extra costs of BEF 100. To use ProWesS and any of our other packages, you need a system with at least 2MB of memory. You should have a harddisk although a two disk system will also work. The use of SMSQ/E is strongly recommended for optimal use of ProWesS.

If you are VAT registered (specify registration number) or live outside the EEC, the amount to be paid is the total (including postage) divided by 1.21 (no need to pay too much).

Payment can be done by EuroCheque in BEF, or by VISA, EuroCard or MasterCard. Credit card orders can be handled by phone. For credit card, please specify name of card owner, card number and expiry date.

Postage : Costs of postage and packaging have to be added. You can choose the quality. Rate depends on no of programs.

copies	*priority mail* Belgium	Europe	World	*ordinary mail* Belgium	Europe	World
one	100	200	240	100	120	145
two	135	340	420	135	190	230
3 or 4	160	560	770	160	310	395
5 to 8	185	870	1250	185	550	705
more	295	1130	1610	295	800	1030

All prices are in BEF, including 21% VAT

Haachtstraat 92, 3020 Veltem, Belgium
tel/fax +32 (16) 48 89 52
joachim@club.innet.be

True Confessions - The Reply

Wolfgang Uhlig

In the last issue of QL TODAY the problems of Qpac2, the hotkey system and the question of how to write a good boot program came up again.

Geoff Wicks wrote a very personal article about button maniacs and his own "true confessions". He asked people to convince him and other non-button-users of why it would be better to use buttons.

That gave me the idea to write a small article dealing with my own boot program and describing why I love to work **WITH** buttons. Both parts belong to each other and have one basic and important theme: the POINTER ENVIRONMENT together with the HOTKEY SYSTEM.

THE BOOT PROGRAM

My boot program has had more or less the same design since I've had a QXL card together with SMSQ(/E) as operating system. Of course little changes take place every now and then, new software comes, an old program has to disappear. Here some parameters must be changed, there a small – at this moment helpful – HOTKEY has to be added, but the frame remains the same.

As I neither work with other operating systems nor on other platforms, all testing and questioning in the boot program can be omitted. I work on my QXL with SMSQ/E and that's it!

The above is, except for some minor details, my boot program. Of course the different programs and extensions are to be found in different subdirectories but I left those out for ease of reading. The names of the Qpac2-items are a little bit different in the German and English versions (information for German readers).

If you delete all the REMarks, from which some people, as I hope, can learn a little, a compact and easy-to-read (in my opinion) boot program remains, that will fit the needs of many a QL-user.

And there we are, with Geoff Wicks the 'button-hater' I find it rather bold – I must say – to judge the character qualities of people by whether they are using buttons or not, as Geoff does. In my opinion this matter is more complex and has to do with the very individual manner of working with one's computer.

There are people who prefer to do no more than two things at the same time. They hate it when two or more jobs have their windows on the screen. Everything has to be clear and tidied up and nothing may cause distraction. The other extreme is people who cannot have enough jobs running on the screen, whomust have everything available at any time. They hate every second they lose searching for a file or task anywhere in the system. A lot of people like to work with the keyboard, mainly those who write a lot. They aremostly able to type with ten fingers, they know dozens of macros or altkeys by heart and feel distracted when having to take the mouse. On the contrary people who draw or paint a lot, who can only write with two or three fingers, love to work with the mouse. There are people who have difficulty with learning to read and write in the right way and are very thankful for being able to choose an item – which is written correctly – with the mouse.

There are more examples of extremes and there are lots of people who are anywhere in between.

There is no "good" or "bad" method or manner and one is no luddite or heretic if one feels better in the one or the other direction!

Going through my boot program you will notice that I belong to those who like to work with buttons, though not to such an extreme extent that I described above. I will try to explain why this is so, giving an example. For a few months I've been busy writing a rather big program in SBASIC which is now ready and will be used for the administration of a bicycle shop here in Nijmegen where I live.

During the development of such a piece of software you do not only learn a great deal by constantly trying out different ways to solve a problem but also do you have new ideas each and every day (okay, I have): "here I want to have another sprite", "oh, now I must write a small extension for a certain purpose", "..I could put a LOGO into all of the faxes, couldn't I?", "so many commands for my HP Deskjet, couldn't I write a printer driverin Basic?",..... and so on! So I very often have to start a new program, a second or third BASIC Job or to wake another program up.

For the development of my program I need the following things – not always ALL of them at the same time, but most of them.

- QD with SBAS/QD-Thing for development and testing
- QD with QBASIC-Thing for compilation

- QD in a patched version with dutch menu-items
- QD to look at and alter different data-files
- Text87 to write documentation and a manual
- EASYMENU to design menus, sometimes twice to compare them
- EASYSPRITE to make sprites
- APPMAN to handle sprites and menus
- DBAS_PTR_EXE, a database program (PD), often two or three times to compare and relate databases
- QFAX to try out, whether this part will function
- QTPI to exchange data between the bicycle shop and my home and therefore

also ..

- ACP - the packer program of Thierry Godefroy
- SBASIC to try out or trace parts of programs, and last but not least ..
- the file menus win_, ram_ und flp_, which are used regularly.

I must say, I'm completely unwilling to spring from one job to another by using the CTRL C "Hotkey".

Geoff also discusses a 'cluttering up' of the screen with irrelevant information. In my opinion this actually happens only when you are multi-tasking jobs which don't take the whole screen and you don't have a chance to hide them anywhere!

So just BECAUSE I like a tidy screen, buttons are of so much importance to me: If I don't want to work in a certain program, I simply drop it. (What else are the famous 'Zzz' buttons for, then??) It will decently find a place in the button frame and wait for another call to continue its work. The button frame thus holds a lot of absolutely NOT irrelevant information and makes it possible to get what you want, in a minimum of time.

Geoff says that he wants to move from one program to another with a minimum of key-presses and he gives an example for this. But where is the advantage of:

a) calling up the 'hotkey' menu, scrolling through it to find the right program and then click on it to start it up (= at least 4 keypresses)

against

b) use the mouse to place the pointer on the button of the program and click with the right mouse button (= 1 'key'press)

???

Most importantly, you really don't HAVE TO always use buttons, only because you have them! Probably everyone has got the one or the other favourite key or key combination for certain tasks. So do I. It would be silly to go through the whole procedure of picking the button frame, starting the hotkeys menu only to start 'QWATCH', which is hidden under, let's say, text87. Of course ALT 'u' is ultimately faster!

The great advantage of the hotkey system actually is the FLEXIBILITY it gives you to find the best way for what you want to do.

As you can see in the above boot program, it is very simple to create buttons for your favourite software, if you have done a lot of HOTKEY-definitions anyway. There are people who do and there are people who don't. All of them are content with what they do and that's the way it should be. The one who tries to conclude a "deep-rooted insecurity" out of this, in my opinion tells more about himself than about others.

Geoff, dat slaat gewoon nergens op! (Geoff Wicks will understand this)

ALT F1 = Wolfgang Uhlig Delistraat 80 NL - 6524 KS Nijmegen

```
100 REMark ***** example boot program for SMSQE
110 REMark ***** First some defaults
120 PROG_USE win1_
130 DATA_USE win1_
140 DEV_USE 1,WIN1_my_favorite_subdirectory_
150 DEV_USE 2,WIN1_another_one_
160 :
170 REMark ***** now the unmissable things
171    REMark ***** If you don't have SMSQE you will have to 'lrespr' here:
172    REMark ***** ptr_gen, wman and hot_rext!
180 LRESPR Qpac2
190 LRESPR Menu_rext
200 LRESPR QLib_run_336mod
210 LRESPR qlib_bin
220 LRESPR History1v26c_cde
230 his_def #0,50
240 LRESPR EASYPTR_ptrmenr_cde
```

```
250 LRESPR qbasic_rext
260 LRESPR FileInfo2_bin
270 :
280 REMark ***** two extensions for database programming
290 LRESPR DBAS_DBAS_BIN
300 LRESPR DBAS_DATA_BIN
310 :
320 REMark ***** bigger letters on a QXL 800x600 screen
330 adr1=RESPR(875):adr2=RESPR(875)
340 LBYTES tall_1_qls,adr1
350 LBYTES tall_2_qls,adr2
360 CHAR_DEF adr1,adr2
370 :
380 REMark ***** ask for some patience
390 CLS #0
400 PRINT #0, "still a few seconds to go, please wait"
410 :
420 REMark ***** now define all my HOT_KEYS
430 REMark ***** a HOTKEY definition consists always of:
440 REMark ***** 'ERT HOT_' and one out of the 'LOAD'-'THING'-'RES'-family, and the
450 REMark ***** parentheses in between you (first) choose the 'key' for (second) the job it will do)
460 REMark ***** if you take a small letter the HOTKEY will function with small letter AND
470 REMark ***** with capital letter. If you take a capital letter, it will function
480 REMark ***** ONLY with capital letter
490 :
500 REMark ***** first the QPAC2 features
510 ERT HOT_THING ('f','files')
520 ERT HOT_WAKE ('p','Pick')
530 ERT HOT_WAKE ('w','Wake')
540 ERT HOT_WAKE ('1','rjob')
550 ERT HOT_WAKE ('h','hotkeys')
560 REMark ***** the three next are very useful and make the system extremely handy
570 REMark ***** the first will bring JOB 0=BASIC on top
```

```
580 ERT HOT_PICK ('b','')
590 REMark ***** next one rebuilds buttonframe and brings pointer to first button
600 REMark ***** thus you are able to find back ANYthing at ANY moment!
610 REMark ***** you can configure QPAC2 to do the same with click on both mouse buttons!
620 REMark ***** just start Menuconfig or Config and go through the configurable features
630 REMark ***** of QPAC2. There is no chance of missing it!
640 ERT HOT_THING ('.','button_Pick')
650 REMark ***** the third one makes EVERY job falling asleep (yawn!)
660 ERT HOT_THING ('z','button_sleep')
670 :
680 REMark ***** now some programs that have to be resident (QD if you want
690 REMark ***** to use the SBAS/QD-thing) or which I want to be resident because
700 REMark ***** of the high-speed access (small programs of my own in this case)
710 ERT HOT_RES ('u','qwatchqx1_obj')
720 ERT HOT_RES ('û','QD')
730 ERT HOT_RES ("{","util_obj")
740 ERT HOT_RES ('d','wolledatum_obj')
750 ERT HOT_RES ('!','wollewecker_obj')
760 :
770 REMark ***** Here they finally come: my favorite programs
780 ERT HOT_LOAD ('æ','EASYPTR_easysprite_exe')
790 ERT HOT_LOAD ('Æ','EASYPTR_easymenu_exe')
800 ERT HOT_LOAD ('ö','EASYPTR_appman_obj')
810 ERT HOT_LOAD ('x','DBAS_DBPTR_exe')
820 ERT HOT_LOAD ('ô','macro_exe')
830 ERT HOT_LOAD ("ò","LDESboot_obj"):REMark ***** a Linedesign booter, not LD itself
840 ERT HOT_LOAD ('ù','QD_lib'):REMark ***** QD with QBASIC as a THING
850 ERT HOT_LOAD ('T','text87plus4')
860 ERT HOT_LOAD ("Ö","menuconfig_german")
870 ERT HOT_LOAD ("ì","QLFED_obj")
880 ERT HOT_LOAD (" ","QTPI_qtpi155_exe")
890 ERT HOT_LOAD ("}",'ACP_obj')
900 ERT HOT_LOAD ("à","Qspread;'\my_favorite_spreadsheet'")
910 ERT HOT_LOAD ("L","qlib_obj")
920 :
930 REMark ***** perhaps you wonder why I take so many of these 'impossible-to-remember-keys'
940 REMark ***** Actually, I DO NOT HAVE to remember them because they will have a button. The
950 REMark ***** advantage is: nearly all my 'normal' letters on the keyboard are free for
960 REMark ***** temporary ALTKEYS or HOTKEYS during a session and I can remember them easily then!
970 :
980 REMark ***** ALT/F3 starts an SBASIC and then "does" everything you wrote in 'myprog_bas'
990 ERT HOT_THING (CHR$(240),'sbasic';'do myprog_bas')
1000 :
1010 REMark ***** some examples of HOTKEYS
1020 REMark ***** write my adress with ALT/F1 in every program where text input is possible
1030 ERT HOT_KEY (CHR$(232),'Wolfgang Uhlig','Delistraat 80','NL - 6524 KS Nijmegen')
1040 REMark ***** ALT/F4 brings up a file_select window in SBASIC
1050 ERT HOT_KEY (CHR$(244),'outl:load file_select$(,,,,,,,1,)',',')
1060 REMark ***** ALT/F5 (in QD German version) sets a marker, deletes line numbers and goes back to
marker
1070 ERT HOT_KEY (CHR$(248),CHR$(240)&"MM"&CHR$(32)&CHR$(240)&"ZL"&CHR$(240)&"MG")
1080 REMark ***** ALT/F6 the same with inserting linenumbers
1090 ERT HOT_KEY (CHR$(234),CHR$(240)&"MM"&CHR$(32)&CHR$(240)&"ZE"&CHR$(240)&"MG")
1100 :
1110 REMark ***** the last HOTKEY is an example for HOT_CMD: wherever you are, this HOTKEY will
1120 REMark ***** bring Job 0, BASIC, on top, write down the 'command' and ENTER it.
1130 REMark ***** don't ask me why QFAX would not go with HOT_LOAD?!
1140 ERT HOT_CMD (CHR$(250),'ex qfax;"-W 2"')
1150 :
1160 REMark ***** Okay, now let's begin to build the button frame
1170 REMark ***** I take the 'EXEP' command for the files-menus because they are then jobs and
1180 REMark ***** a job can simply be removed if something goes wrong
1190 REMark ***** the parameters here are: '\b' = colour combination, '\n' = the name YOU want
1200 REMark ***** to give to the button and '\I' = contents of the device
1210 REMark ***** there are more parameters possible, but these do best
1220 EXEP "files";'\b 3 \n WIN1 \I win1_'
1230 EXEP "files";'\b 3 \n WIN2 \I win2_'
1240 EXEP "files";'\b 3 \n RAM1 \I ram1_'
```

```
1250 EXEP "files";'\b 3 \n RAM2 \I ram2_'
1260 EXEP "files";'\b 3 \n FLP1 \I flp1_'
1270 EXEP "files";'\b 3 \n CD \I win3_':REMark if the CD-Rom is on ›E: on your PC
1280 :
1290 REMark ***** now the buttons which belong to the system. I must confess that I still don't
1300 REMark ***** understand which exactly the differences are between BT_SLEEP, BT_WAKE, BT_HOTKEY
1310 REMark ***** and BT_EXEC but the way I've arranged it works best, so what?
1320 REMark ***** here, too, you could give a name of your choice, for
1330 REMark ***** example ‹BT_SLEEP 'wake','Wake up, little Suzie'› or something like that
1340 REMark ***** You need no parentheses here!
1350 BT_SLEEP 'Wake'
1360 BT_SLEEP 'Pick'
1370 BT_SLEEP 'Rjob'
1380 BT_SLEEP 'Jobs'
1390 BT_SLEEP 'channels'
1400 BT_SLEEP 'hotkeys'
1410 BT_SLEEP 'sysdef','MAKE_DIR'
1420 REMark ***** the buttons for the most important programs which of course have to be
1430 REMark ***** HOT_LOADed or _CHPed or _RESed
1440 REMark ***** you take the key you defined above and give it a name of your choice, that's it
1450 BT_HOTKEY 'û','QD-BASIC'
1460 BT_HOTKEY 'ù','QD-LIB'
1470 BT_HOTKEY 'à','QSPREAD'
1480 BT_HOTKEY 'T','T91'
1490 BT_HOTKEY 'ò','LDES'
1500 BT_HOTKEY "Ö","CONFIG"
1510 BT_HOTKEY 'æ','SPRITE'
1520 BT_HOTKEY 'Æ','MENU'
1530 BT_HOTKEY 'ö','APPMAN'
1540 BT_HOTKEY 'x','DATABASE'
1550 BT_HOTKEY "}","PACKER"
1560 BT_HOTKEY "ì","QLFED"
1570 BT_HOTKEY " ","QTPI 1.55"
1580 BT_HOTKEY "{","UTIL"
1590 REMark ***** and now the command that will make everything work:
1600 HOT_GO
1610 :
1620 REMark ***** still some more things to do
1630 EXEC freemembt :REMark ***** small program which fits into the button frame and
1640                 REMark ***** shows free system memory
1650 HOT_DO "u"     :REMark ***** Hot_"does" the watch from above
1660 DISP_UPDATE 2,2:REMark ***** important for the QXL
1670 IO_PRIORITY 10 :REMark ***** the same
1680 :
1690 REMark ***** two keyboard autorepeat changes
1700 POKE_W 163980,20:REMark ***** waits only 20 (what, ms??) before autorepeat starts
1710 POKE_W 163982,1 :REMark ***** makes autorepeat extremely fast (...,5 is slow)
1720 TRA 3 :REMark ***** only SMSQE! sets tra table to IBM
1730 EX win1_util_sysmon_exe :REMark ***** important utility
1740 HOT_DO '.' :REMark ***** see above, brings pointer to first button
1750 :
1760 REMark ***** a procedure to bring me to the normal QL-resolution and play Lonely Joker ;)
1770 DEFine PROCedure qql
1780 DISP_SIZE 512,256
1790 WMON:PAUSE 25
1800 EX LJ_exe
1810 END DEFine qql
1820 :
1830 REMark ***** and back to work :(
1840 DEFine PROCedure svga
1850 DISP_SIZE 800,600
1860 END DEFine
1870 REMark ************************************************************************
```

■

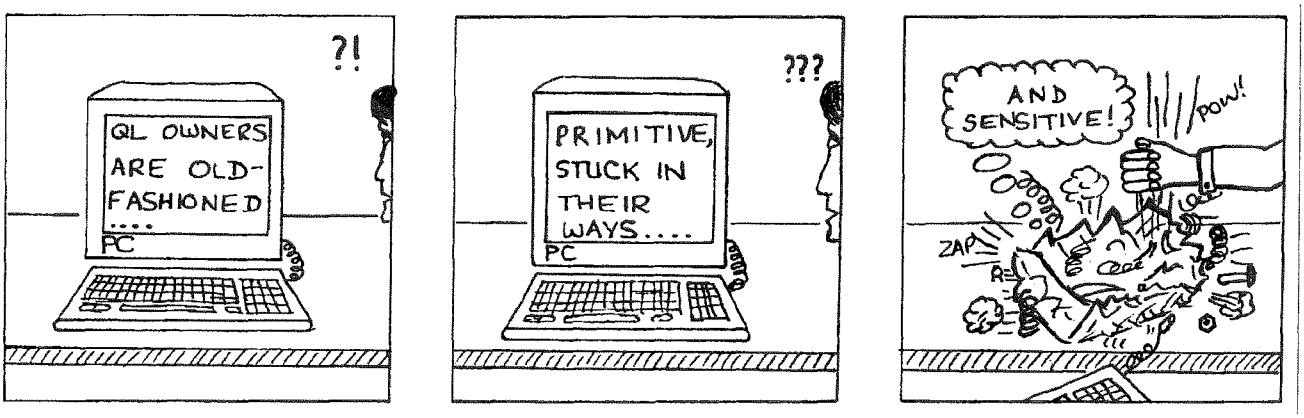
Chameleon Sysmon

Jochen Merz

A number of customers already asked me if and when QPAC 1's very useful SYSMON program will be updated so that it is colour configurable. They would like to see it appearing red in the button frame instead of green, so that it fits into the general appearance of their button frame. I never worried about the colour myself, but I prefer a green button frame. As we get plenty of this kind of request for all kinds of changes in all the products, and we can't, unfortunately, afford to pay 500 programmers to fulfil all the wishes at once as you can imagine, the chances that this change in SYSMON will happen soon are quite small. However, not much work is required so that you can do it yourself if you like. Type in the following program and it will patch your SYSMON to red. As the program will use some advanced features of SMSQ's SBASIC, here a quick explanation so that people who do not use SMSQ can convert it using old-fashioned SuperBASIC commands instead. First, SYSMON is copied into a separate file - never patch the original! The file SYSMON_red is then searched for two patterns which it recognises, so that the patch will work independently of version and language. The first patch makes sure that free area in the sprite is printed in red instead of green. The strange MODE 4 display organisation is quite helpful this time: every even byte represents green, every odd byte represents red. Therefore, all we have to do is add 1 to the address in which the pixel is set - this makes it red instead of green. The next occurrence patches the border - all we have to find is the window definition, in which we patch the real colour green (4) to red (2).

```
100 REMark Change SYSMON to red colour
110 REMark *** will run on SMSQ only! ***
120 INPUT 'Sysmon filename >';file$
130 filered$=file$&'_red'
140 COPY file$ TO filered$
150 OPEN #3,filered$
160 l=FLEN(#3):DIM s$(8)
170 FOR c=0 TO l-10 STEP 2
180  LGET #3\c,chk1,chk2
190  IF chk1=$6E1A4A87 AND chk2=$6D0605F2
200   WGET #3\c+8,grn:IF grn=$C800:WPUT #3\c+8,$C801:PRINT 'Red contents patched'
210  END IF
215  IF chk1=$184000C AND chk2=0
217   WGET #3\c+12,grn:IF grn=4:WPUT #3\c+12,2:PRINT 'Red border patched'
218  END IF
220 END FOR
230 CLOSE#3
240 PRINT 'Finished'
```

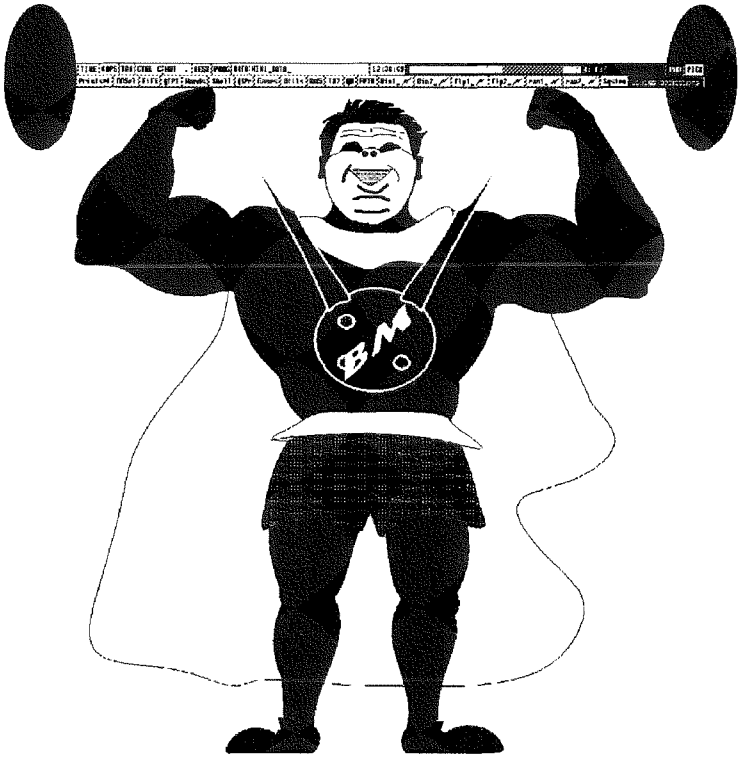
■



Button Up Your Overview

Roy Wood

I was leafing idly through the last copy of QL Today when I noticed a gauntlet thrown down by a man who had just stopped short of table dancing. Could this belong to the fearless Geoff Wicks I thought, he who uses Just Words and no buttons? Never being one to pass up a good challenge I leapt into the fray, Up up and away 'Button Man'. Well I had to reply, didn't I, because he made such a fuss with his buttonless fly remarks. Yes I have a lot of buttons on my screen and, now I have an Aurora, I have room for even more but they are not there just for show. I use them all at some point in a busy day and if I had to 'control C' through them to find the program I want then my day would be even longer. You see not everyone has only bought three programs in his computing career (no wonder us traders and software writers are so poor). If I only had three programs running I would probably not bother with a button frame either. The trouble is that I do a lot with my QL that involves the interaction of more than one program so the buttons are handy. Poor old Geoff has not realised that you can have a hotkey to pull up the button frame and select the program you want from there. I make that two keypresses and if you want to 'control C' to the third program in a group then I am already one up on him. (Add it up for yourself - I am sure you will Solvit) There are some unenlightened programmers who have not used Jochen's excellent QMenu extensions to give us access to a files menus so, if I am using one of their programs, I just press the centre mouse button, click on the button for the files directory I need, highlight the file and click on the progam underneath putting me in a position to use the stuffer buffer to stuff the file name. So thats four buttons accounted for - flp1_, flp2_, win1_, win2_, ram1_, ram2_ - one for each root directory. Multibutton is very useful (another Jochen Merz invention) because it gives a readout of the free memory, a button to click on for the time and date, as well as allowing you to change the DATA_USE, PROG_USE and DEST_USE devices from a menu. This is an invaluable tool for those programs which insist on putting things into these devices and not allowing a tidy subdirectory structure. It is all one button but it looks like 10, count it how you like. Now we have Sysmon from - guess who? This shows you at a glance how much your memory has become fragmented and what each program is using as well as keeping an eye out for memory corruption and wailing at you to tell you when it happens. Save a file to your hard disk when this little boy is howling and you risk a map corruption. Of course I also have the notepad from the same QSUP package for quick notes and quick access to the scrap function and the Pick, Exec and Rjob menus from QPAC II because all of those are useful. From Pick you can get to any job in the machine, Exec will take you to any executable thing or any of the other QPAC 2 utilities and Rjob will remove any offending program that has crashed. With me so far? That is 15 buttons if we include Multibutton as 10. Next there are three buttons for my own jobs. DSel, Printsel and Words. Dsel will call up a Qmenu menu of my DATAdesign databases and allow me to load a datadesign file quickly (address book, QL Today subscription list, suppliers, Invoices all to hand in seconds - try that from Archive!). Printsel will run one of four programs for producing the Q Branch, Miracle Systems invoices, envelopes etc. - I need that a lot too. The last one 'Words' can call up three



different versions of QD, the Qtyp II dictionary editor, and two of Geoff's own programs - Thesaurus and Style Checker. You may notice here that I could have a button for each of these items separately but I am not a slave to fashion - why use 501s when you have a zip available ? (do I need a Style Check?). There are also buttons for FiFi (for finding files lost in the system) and the amazing QTPI used daily for comms functions (Geoff does his on the PC - snigger). Next comes 'Shell' to call up Albin Hessler's Cueshell program for copying purposes and disk comparison, QSpr for the QSpread speadsheet and another of my own buttons, Games, to call up a menu of games (for those odd spare moments - usually Lonely Joker or QShang). Utils is yet another of my own Qmenus which gives access to a lot of odd programs that I use occasionally such as Menuconfig, Winlink, ACP etc. (buttons = 24 and counting) Next comes DM5 for Disk Mate 5. I used this to copy our demos, and many other disks. You can copy whole subdirectories from one device to another and examine the disks themselves wth this tool. After that is T87 for Text 87 - used for all word proccessing, QD for programming and the sending of faxes via QFAX and finally EPTR which is yet another of my own menus to call up the various parts of Easyptr for when I am writing programs. That is the end of the normal progam buttons and they are followed by the files buttons I mentioned earlier and the two ProWesS buttons, utilities and applications that give me access to the ProWesS programs, 38 buttons in all and most of them get a click at least once a day. All of these have hotkeys associated with them and I do sometimes use them to call up the programs that way but the button frame is the quickest and easiest to use. Geoff thinks that studying Jochen's boot file is not the best way to learn how to use it but I disagree. I could not get to grips with QPAC II at first and tried QPACER which asked questions assuming I knew what it was talking about - I didnt. I looked at the manual for QPAC II but got nowhere so I called Jochen and he sent me his boot with explanations and remarks and I was writing boot files in no time. This is the thinking behind the QPAC II Supplement that Q Branch give away with QPAC II. This is a separate disk with a complete boot file that will call up a few buttons and a step by step explanation of what each line does. You can use this boot file as a starting point and play around with the structure to get a feel for how the prgram runs. Your boot file will always evolve because your learn more about the machine each time you use it and you buy new programs (well some people do at least). If you only want to use Xchange and Perfection (Ugh!) then, of course you don't need buttons - you can tie your loincloth up with string. Oh and about those dinosaurs - we knew you could remember them, you've got a Thesaurus. ..... Wilma!

■

The Z88 Sourcebook Review

Darren D. Branagh

The Z88 Sourcebook is a freely distributable Public Domain Disk based book, available from Steve Johnson (Disk number SJPD61, I believe) and written, as the author says "not as a replacement to the Z88 user guide, but as a supplement to it, filling in areas not covered by the user guide."

This is a very accurate description in my opinion. The book is very well written, and the author draws inspiration from a very wide list of publications, such as Update Magazine, Z88 Fax News, Z88 EProm, and other newsletters that have since ceased to be - and are unavailable nowadays, except via this sourcebook.

As the book is written in plain text, The Author gets around the Z88's unusual additional keys by using ‹› (less than, greater than) as DIAMOND, AND [] (brackets) as SQUARE. This makes following the various procedures much easier to follow. The plain text format is also handy in that it is easily portable to most Word-processors, making printing of a Hard Copy virtually painless, should you wish one - It is also easy to download the entire thing to the Z88 and store it there instead!!

There is a vast amount of additional information, for instance, did you know that the Z88 has variable sound capabilities? No? Then try one of these, from the BASIC command line:

```
VDU 1,52,33,38,34,34
VDU 1,52,33,34,33,34
VDU 1,52,33,40,33,33
```

I found it amazing that something so significant gets little or no mention in the user guide. However, the Z88 sourcebook goes into sufficient detail on many such topics. It lists all the

Screen control codes (i.e. codes that are used to control the screen and to print special characters not found on the keyboard) and gives a full section to Pipedream Tips and Tips on using BBC BASIC. The Pipedream Tips includes how to achieve multiple columns easily, insertions, deletions, separating and moving text, block moving, general editing, and much more.

For instance, on the subject of its Graphics capabilities:

Quote: "The Z88 is capable of producing IBM-style line graphics. These are again generated by VDU commands in the following format:"

```
VDU 1,ASC("2"),ASC("*"),ASC("char")
```

where char is a letter in the range A through to O. Here is an example program for the Z88:

```
10 Z=65
20 REPEAT
30 VDU 1,ASC("2"),ASC("*"),Z
40 VDU 9
50 Z=Z+1
60 UNTIL Z=80
```

I'll bet many of you didn't know that!!

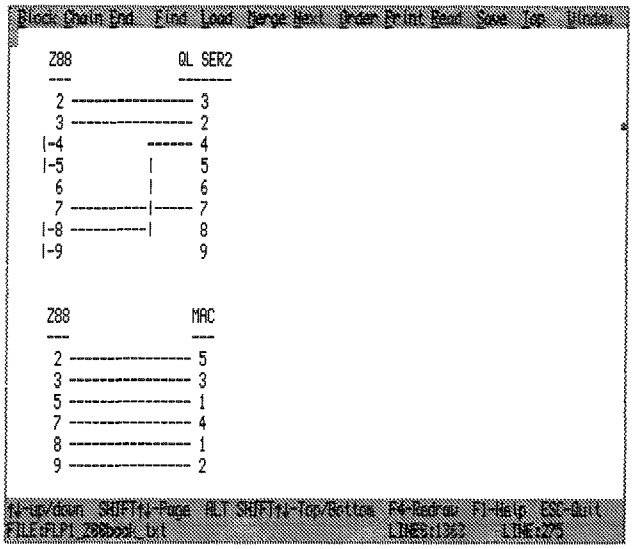
Among the other many "new discoveries" was the fact that the Z88 has the ability to have up to 64 UDG's (user definable characters) a facility I recently used to create a sprite for a game I wrote, and that BBC BASIC on the Z88 comes with an inline assembler built-in! According to the book, the variable P% is used as a program counter, and the user must set P% to the desired start point for the machine code before envoking the assembler. The assembler can be envoked by the "[" symbol, and unenvoked using "]". Below is a sample program:

```
10 DIM code 100
20 P% = code
30 [
40 LD BC,50
50 RET
60 ]
```

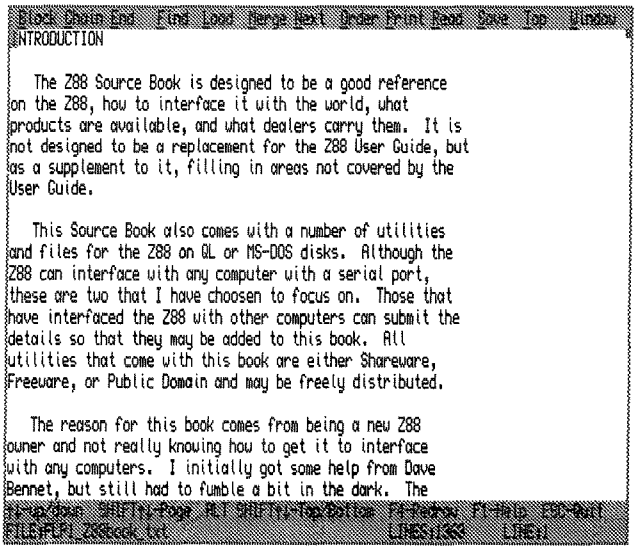
To quote the Author: "It is recommended that the user have a good knowledge of Z80 machine code programming before trying the assembler, as lock up on your Z88 could cause it to do a hard reset to a "virginal" blank state."

Wise words better told than learned the hard way!!

The author also goes into the ins and outs of Memory Organisation (addresses, etc), available additional hardware and software (including Mini-Reviews), Rampacks and EProms, Batteries, General Care and Cleaning of the Z88, Online Resources, and Technical Specifications.



The book is simply full of useful pieces of information like this; everything from pipedream to importing and exporting to either a QL or an IBM PC is covered, in step by step detail. The author also gives full details of most of the various cables needed to connect the Z88 to the outside world - this includes the pin-outs and type of cable needed, i.e. full details on how to connect a Z88 to a PC (DB25 connector), a QL, an Apple Mac, and another Z88 are included.


INTRODUCTION

The Z88 Source Book is designed to be a good reference on the Z88, how to interface it with the world, what products are available, and what dealers carry them. It is not designed to be a replacement for the Z88 User Guide, but as a supplement to it, filling in areas not covered by the User Guide.

This Source Book also comes with a number of utilities and files for the Z88 on QL or MS-DOS disks. Although the Z88 can interface with any computer with a serial port, these are two that I have choosen to focus on. Those that have interfaced the Z88 with other computers can submit the details so that they may be added to this book. All utilities that come with this book are either Shareware, Freeware, or Public Domain and may be freely distributed.

The reason for this book comes from being a new Z88 owner and not really knowing how to get it to interface with any computers. I initially got some help from Dave Bennet, but still had to fumble a bit in the dark. The

Using this information I was able to solder up a cable to connect my QL to it - using an old joystick connector and a length of cable - in only about 10 minutes, saving myself a few quid to boot! When used in conjunction with Phil Borman's IMP/EXP program, and Dilwyn Jones's Archive<->Pipedream software (Both of which are also PD from Steve Johnson) I have all I need to transfer data between the QL and the Z88 - and cheaply, too!!

On the Negative side however, the book mentions various utilities and programs that were originally included on the disk with it. Unfortunately, these are missing from the version I have, and I can only assume they have somehow been "separated" over the years, and therefore lost, prior to its inclusion in the PD Libraries in the UK, as the book originated in America some years ago. This is a very minor gripe, as these would only have been a minor plus to what is already an excellent book.

For those of you interested in obtaining Z88 PD software, send £1.50 (plus P&P) to:

Mr. Ian Braby, Z88 Software Library, 1 Butts Cottages, Copse Road, St. Johns, WOKING GU21 1SU. ENGLAND.

and he will send you a comprehensive 30 page catalogue, detailing some 230+ programs available on EProm or QL disk for the Z88. The entire Library is available for £15!!

However, I'm digressing. The Z88 is an excellent portable computer (especially when used with a QL) and as a result, deserves a decent user guide. With the Z88 sourcebook, this has become a reality - and as it's as good as free, no self-respecting Z88 user should be without it!

■

More Words

Jochen Merz

I hope the title of this article does not confuse you, but in the end that's what this article is all about: just words! It describes how you can extract all the words from a QTYP dictionary into an ASCII-file.

A number of customers asked me how to do it, and I promised to write an article. There may be various reasons why somebody wants to extract the complete word list: if you wish to edit a dictionary, you may find that it is easier to do it with an editor like QD. Also, this is the only way to merge two dictionaries (i.e. expand one dictionary into an ASCII wordlist and merge it into the other one).

The program which does it for you is written for SBASIC in SMSQ - it requires some minor modification to run it under SuperBASIC in QDOS. It uses two function of the menu extension - FILE_SELECT$ to read the dictionary filename and REPORT_ERROR to write out an error and wait for confirmation. Naturally, the QTYP_SPELL extension has to be loaded otherwise you could not access QTYP dictionaries at all!

First, a window is set up. Next, a dictionary filename is read in. SPELL_CLEAR clears any existing dictionary from the memory, and the selected dictionary is then loaded using SPELL_NEW.

The wordlist is created using the dictionary filename, with _words appended. If this fails, the program stops. Make sure the total filename length does not exceed 36 characters if this happens.

The string begin$ contains all the characters with which every word in the dictionary may begin. In the example, the German Umlauts "äöü" have been sorted into the alphabet at the right position. This can easily be changed for other languages.

Then, every character is put into QTYP's comparison buffer in turn and SPELL_WORD$ is repeatedly being read until an empty string is found, which means, that there is no matching word with this starting character anymore.

This is how you get a complete ASCII file filled with all the words contained in a dictionary. You create a new dictionary or add it to an existing dictionary using the dictionary editor QTYP_DED.

I have just discovered that, at the time I write this article, there seems to be a problem if you try to run this program on QPC - we're working on it. We don't know yet whether it is a bug in the SPELL Extension or in QPC. By the time you read this article, it may be fixed. Just check the version numbers in the JMS ad or check the News section.

```
100 IF DEVTYPE(#0)<>1:OPEN#0,con
110 OUTLN#0,512,256,0,0:IF DEVTYPE(#1)<>1:OPEN#1,con
120 WINDOW 512,256,0,0:BORDER 1,7:PAPER 0:CLS
130 CLOSE
140 :
150 REPeat
160  dic$=FILE_SELECT$('Select Dictionary',,,,,,,3)
170  ERT SPELL_CLEAR
180  dic=SPELL_NEW(dic$)
```

```
190  IF dic>0:EXIT
200  REPORT_ERROR dic,,,3
210 END REPeat
220 :
230 words$=dic$&'_words'
240 words=FOP_OVER(words$)
250 IF words<0:REPORT_ERROR words:CLOSE:STOP
260 :
270 ct=0
280 begin$='aÇbcdefghijklmnoäpqrstuçvwxyz'
290 FOR c=1 TO LEN(begin$)
300  AT 22,0:PRINT \\\begin$(c)
310  ERT SPELL_CHECK(#dic,begin$(c))
320  REPeat
330   word$=SPELL_WORD$(#dic)
340   IF NOT LEN(word$):EXIT
350   PRINT#words;word$;
360   ct=ct+1:AT 24,8:PRINT ct
370  END REPeat
380 END FOR
390 CLOSE
400 PRINT \\'Done ...'
```

■

Snippet's Corner - Part 4

M. Knight

Some QL programs use the real-time clock for timings and some use the file update dates present in file headers. In these cases it is a good idea to permit the user to check and set the clock, because software that has crashed a QL badly can corrupt the clock too. The routine provided here allows users to both check and set the clock at the same time, without affecting the time if it doesn't need changing.

Many QL clock setting routines expect the user to type in the SDATE parameters which is a very error-prone method of setting the clock; one unnoticed error can put the clock out by years. The advantage with this routine is that the current setting is visible all the time and the user need only use the cursor keys. If you have a Gold Card or Super Gold Card it is a good idea to use the PROT_DATE 1 setting when experimenting with this routine to protect the current setting. Press the reset button when you have finished to leave the clock unaltered on these systems.

Those without the Turbo Toolkit but using SuperToolkit II should replace the CURSOR_ON command with the appropriate line as follows:

```
30800 CURSEN#0
```

If you are using Turbo to compile a program with this routine in then the variables Tk_Count and Tk_Choice can be defined as integers using the compiler directive:

```
IMPLICIT% Tk_Count,Tk_Choice
```

...somewhere near the start of the program and with a suitable line number, while the equivalent for Q-Liberator users is:

```
DEF_INTEGER Tk_Count,Tk_Choice
```

Q-Liberator users should examine chapter 14 of the manual for advice on the use of DEF_INTEGER and the $$i directive in Q-Liberated programs. Turbo users wishing for more detail on the use of IMPLICIT% and integer constants should look at chapters 3 (section 3.3.2.10) and 4 (sections 4.1 to 4.1.2.4).

Set_CLOCK is notable in the Snippet's Corner collection as it is the first interactive routine. It will respond to the F4 key which is the standard QL redraw key for non-pointer programs, and if you insert a suitable call into the listing after the REMark it works very smoothly. Set up a routine called something like Redraw_SCREEN and call it instead of CLS. Notice that the main loop is kept very small to keep it responsive and is only exit when the user presses a valid key. There is proper use of SELect and the REPeat structure to keep the listing compact and functional.

Notice that I have avoided KEYROW and used only INKEY$ for compatability reasons. Some CST Thor models have no KEYROW command

or a buggy version of it and also on some QL systems KEYROW is buggy or variable in speed. QL programs using KEYROW also respond at differing speeds depending upon the speed of the processor they are running on.

This means programs which are fine on a normal QL would be awkward on a Gold Card because the keyboard response is too fast and on a Super Gold Card or a QXL they would be more or less unusable. INKEY$ responds at a fairly fixed speed on any QL or SMSQ system.

Listing 4.

```
100 REMark programmer must check that CONsole WINDOW attached to #1 is large enough.
110 CLS
120 Set_CLOCK 1
130 :
30720 DEFine PROCedure Set_CLOCK(Tk_Channel%)
30725   LOCal Tk_Count,Tk_ShowLoop
30730   LOCal Tk_ClockLoop,Tk_Choice
30735   LOCal Tk_PtrVal%,Tk_UpdateSize(7)
30740   LOCal Tk_ClockPointer$(7)
30745   DATA 1, 19, 60, 16, 3600, 13, 86400
30750   DATA 10, 864000, 9, 2.4192E6, 6
30755   DATA 3.1536E7, 3, 3.1536E8, 2
30760   Tk_ClockPointer$=""
30765   RESTORE 30745
30770   FOR Tk_Count=1 TO 7
30775     READ Tk_UpdateSize(Tk_Count),Tk_PtrVal%
30780     Tk_ClockPointer$=Tk_ClockPointer$&CHR$(Tk_PtrVal%)
30785   END FOR Tk_Count
30790   Tk_PtrVal%=1
30795   CLS#Tk_Channel%
30800   CURSOR_ON#0
30805   Clear_BUFFER 0
30810   REPeat Tk_ClockLoop
30815     REPeat Tk_ShowLoop
30820       AT#Tk_Channel%;0,0
30825       PRINT#Tk_Channel%;DATE$&" "&DAY$
30830       AT#Tk_Channel%;1,CODE(Tk_ClockPointer$(Tk_PtrVal%))
30835       PRINT#Tk_Channel%;"@"
30840       Tk_Choice=CODE(INKEY$(#0))
30845       IF Tk_Choice<>0 THEN EXIT Tk_ShowLoop
30850     END REPeat Tk_ShowLoop
30855     AT#Tk_Channel%;1,CODE(Tk_ClockPointer$(Tk_PtrVal%))
30860     PRINT#Tk_Channel%;" ";
30865     SELect ON Tk_Choice
30870       =10
30875         CLS#Tk_Channel%
30880         RETurn
30885       =192
30890         Tk_PtrVal%=Tk_PtrVal%+(Tk_PtrVal%<7)
30895       =200
30900         Tk_PtrVal%=Tk_PtrVal%-(Tk_PtrVal%>1)
30905       =208
30910         ADATE Tk_UpdateSize(Tk_PtrVal%)
30915       =216
30920         ADATE -Tk_UpdateSize(Tk_PtrVal%)
30925       =244
30930         CLS#Tk_Channel%:REMark reset the screen here if writing serious software.
30935     END SELect
30940   END REPeat Tk_ClockLoop
30945 END DEFine Set_CLOCK
30950 :
31630 DEFine PROCedure Clear_BUFFER(Tk_Channel%)
31635   LOCal Tk_KeyLoop
31640   REPeat Tk_KeyLoop
31645     IF INKEY$(#Tk_Channel%)="" THEN RETurn
31650   END REPeat Tk_KeyLoop
31655 END DEFine Clear_BUFFER
31660 :
```

■

Non-Repeating Random Numbers

Dilwyn Jones

A while back I needed a routine which would generate random numbers in such a way that the numbers would never repeat themselves during one run. Examples of where such a routine might be needed are card games (where you draw a card from the pack and can't draw that one again), choose random numbers for a national lottery draw, or simply for general use in games where you need to perform an action once at random, but you are only allowed that action once.

I have used a string to store a list of byte values in a given range (here, 1 to 255 or whatever other limit value you enter). Each successive character in the string stores a different number. One of these is pulled out at random, then removed from the string. So by successively choosing a number from the string, then removing it, and continuing the process until the string is empty, we achieve our aim.

This program can only handle numbers up to 255, since that is the limit of the character codes which can be held in a string. Advanced users could store a list of two byte (word integer) or 4 byte (long integer) values and use commands such as MOVE_MEMORY in toolkits to shuffle the list elements around in a manner similar to the string slicing techniques I used.

```
100 REMark non-repeating random numbers
110 CLS : INPUT'How many numbers (1-255) › ';num
120 RANDOMISE
130 random$ = '' : REMark holds the random numbers
140 REMark create "num" random numbers from 1 to num
150 FOR a = 1 TO num : random$ = random$ & CHR$(a)
160 FOR a = 1 TO num
170   REMark choose an element from the list at random
180   element = RND(1 TO LEN(random$))
190   REMark what is the number at that element
200   rand_no = CODE(random$(element))
210   PRINT ! rand_no ; : REMark show the number chosen
220   REMark remove this number from the list, so can't happen again
230   lr = LEN(random$) : REMark length of list
240   IF lr = 1 THEN
250     random$ = '' : REMark remove last element from list
260   ELSE
270     IF element = 1 THEN
280       random$ = random$(2 TO lr) : REMark st start of list
290     ELSE
300       IF element = lr THEN
310         random$ = random$(1 TO lr - 1)
320       ELSE
330         random$ = random$(1 TO element-1) & random$(element+1 TO lr)
340       END IF
350     END IF
360   END IF
370 END FOR a
380 PRINT : PRINT #0,'Program finished.'
```

■

QMAC Review

Norman Dunbar

When Dilwyn asked me to review this assembler development kit I readily agreed as I have used the package for a number of years. However, one thing I did not take into consideration at the time was the fact that it is incredibly difficult to say anything interesting about an assembler – we are into 'anorak country' here !

This package comes originally from GST Computer Systems in Cambridge, but is now only available from Quanta for about £15.00 including postage. The package is being updated by Phil Borman, so it is being maintained and developed, even as we speak.

Recent enhancements are a standard Config block so that you can configure all your favourite options to save you having to type them in each time you want to assemble something; the ability to read from the DATA and/or PROG default devices; a new INCBIN

directive that allows you to include binary files in the final output file (useful for including screen dumps, data files etc) and conditional assembly.

The version reviewed is 1.13 and includes QMAC the assembler, QLINK the linker, a macro library, QED an editor and QED_EXE a slightly different version of the QED editor and a few example programs and lots of include files.

Before you start

If you don't have a clue about what an assembler does, here is a short introduction.

Remember back in the old days when QL World was still around and DIY Toolkit articles were featured every month? Well DIY Toolkit utilities were written in machine code - the language that the 68000 series of processors understand. If you wanted to use a particular utility, then you had a few choices. The first was to wait for the disc to come out and buy it, the second was to type in the SuperBasic listing and run it to produce a code file that could be RESPR'd and CALL'd and the third was to type in the source code and assemble it into a code file on disc. This was then ready to RESPR & CALL as with option two.

The difference is that you have to type in a whole load of hex numbers or a whole load of 'words', I prefer to enter the words rather than the numbers - it makes life easier when your code fails to work properly and you have to debug it. I find it easier to debug words rather than numbers.

So you start with the editor and create a source file, usually this will be saved to disc with an extension of '_ASM'. The assembler is then run and this reads in the source file and converts it to a relocatable file having an extension of '_REL' or '_OBJ' or whatever. At this stage most of what you typed has been converted into numbers, nothing more. The file is not quite ready to run yet as it may need to be linked so the linker is invoked to do this.

The linker simply reads the relocatable file and 'fixes up' any links into library files etc, the final output is your job or utility, ready to run.

So enough of the background, on with the boring stuff !

QMAC - the Assembler

The assembler is a 2 pass one - this means that it scans through the source code twice. The first time builds up a symbol table and the second generates the output file. This file is in SROFF format (Sinclair Relocatable Object File Format) unless you have specified otherwise using the NOLINK command line option. This forces the assembler to produce a file that can be run (EXEC'd or CALLed) directly - however this option reduces the options that you can use in your code. Personally, I have never used this option.

The SROFF file needs to be passed through the linker program, QLINK, before you have a file that can be used. QLINK will be discussed later.

The assembler has a large number of options and I am not going to bore you any more by listing them, take my word for it that there are loads of different combinations of settings that you can use to get listings, symbol tables, errors only or rename the output file, or to prevent the generation of an output file and so on.

There are a number of ways that you can activate the assembler :

Interactive mode - this is when you EXEC QMAC and when it starts up, you are presented with a prompt. You can type in the first file name and options and when done, you will get the prompt again so you can assemble another source file. This continues until you press ENTER on its own as the command line to exit the assembler.

Non-Interactive mode - this is where you EXEC QMAC and give it a command line in quotes to work on. The assembler will load and carry out the processing you have requested and when done, will exit.

Hidden mode - this is where you pass an option of '-NOWINDS' and the assembler simply does its work hidden away from the world. You never see it on the screen and all normal on screen messages are kept hidden from sight. This mode is useful when running the assembler under the control of a front end program like MAKE or similar. (MAKE is on the C68 discs and Jochen sells QMAKE which is a specially written version of the program.)

As a small diversion. If you have ever worked with 'real' programming projects where the source code is in a number of files, then MAKE is a useful front end program to master. It works out which of the source files have been changed and only assembles (or compiles - it depends on the language being used) those. It will optionally run the linker to link the SROFF files into the finished code file ready to test. In addition you can make MAKE do almost

TF Services

superHermes

A major hardware upgrade for the QL

- All Hermes Features (see below for list) PLUS full 19200 throughput on ser1/ser2 not affected by sound
- IBM AT keyboard interface (plus foreign drivers)
- HIGH SPEED RS232 industry standard two-way serial port. 4800cps throughput (supergoldcard - qtpi - zmodem) at 57600bps
- THREE low speed RS232 inputs (1200 to 30bps) Driver for SERIAL MOUSE supplied. Other uses include RTTY/graphics tablet etc
- THREE spare I/O lines (logic) with GND/+5V
- Capslock/scrollock LED connector
- Turbo/keylock connectors
- 1.5k user data permanently storeable in EEPROM

All this on a professional board about twice the size of the 8049 co-processor it replaces

Cost (including manual/software)..**£90** (£92/£87/£90)
IBM AT UK layout Keyboard.....**£22** (£24/£23/£27)
Serial mouse.....**£11** (£13/£12/£14)
Capslock/scrollock LED .....**£1** (£1.50/£1/£1.50)
Keyboard or mouse lead.....**£3** (£3.50/£3/£3.50)
High speed serial (ser3) lead.....**£4** (£4.50/£4/£4.50)

Hermes available for £25 (£26/£24/£27) **(wking ser1/2 and independent input, debounced keyboard & keyclick)**

Minerva

The ORIGINAL system operating system upgrade

MINERVA RTC (MKII) + battery for 256 bytes ram. CRASHPROOF clock & I²C bus for interfacing. Can autoboot from battery backed ram. Quick start-up.

OTHER FEATURES COMMON TO ALL VERSIONS

DEBUGGED operating system/ autoboot on reset of power failure/ Multiple Basic/ faster scheduler- graphics (within 10% of lightning) - string handling/ WHEN ERROR/ 2nd screen/ TRACE/ non-English keyboard drivers/ "warm" fast reste. V1.97 with split OUTPUT baud rates (+ Hermes) & built in Multibasic. 1.98 now out supporting Qshang.

First upgrade free. Otherwise send £3 (+£5 for manual if requd). Send disk plus SAE or two IRCs.

MKI...£40 (£41/£40/£43) **MKII...£65** (£66/£63/£67)

QL REPAIRS (UK only)

Fixed price for unmodified QLs, excl microdrives. QLs tested with Thorn-EMI rig and ROM software.

£27 including 6 month guarantee

I2C INTERFACES

Connects to Minerva and any Philips I²C bus

Power Driver Interface Similar to parallel below (16 I/O logic lines) except that 12 logic lines can be used to control 8 current carrying outputs (source and sink capable)
2 amp (for 8 relays, small motors)... **£40** (£43/£38/£44)
4 amp total (for motors etc) ..... **£45** (£48/£43/£50)
Relays (Eight boxed 3 amp 12v 2-way mains relays connecting to 2 amp power driver.....**£25** (£28/£23/£27)
Parallel Interface Gives 16 input/output lines. Can be used wherever logic signals are required.....**£25** (£28/£23/£27)
Analogue Interface Gives eight 8 bit analogue to digital inputs (ADC) and two 8 bit digital to analogue outputs (DAC). Used for temperature measurements, sound sampling (to 5 KHz), x/y plotting .....**£30** (£31.50/£29/£30)
Temp probe (-40°C to +125°C) .....**£10** (£10.50/£10/£11)
Connector for four temp probes .....**£10** (£10.50/£10/£11)
Data sheets.....**£2** (£2.50/£2/£3)
Control software & manual (for all I/F)....**£2** (£2.50/£2/£3)

QL SPARES

Keyboard membrane..... £12 (£12.50/£12/£13.50)
1377 PAL..... **£3** (£3.50/£3/£4)
Circuit diagrams..... **£3** (£3.50/£3/£4)
68008 cpu or 8049 IPC..... **£8** (£8.50/£7.50/£9)
8301/8302 or JM ROM set..... **£10** (£10.50/£10/£11)
Serial lead..... **£10** (£11.50/£11/£12)
Power supply (sea mail overseas).... **£12** (£17/£16/£21)

Other components (sockets etc) also available

QBBS Bulletin Board

UKs first QL scrolling bulletin board

Megabytes of files. Messages to/from UK/Belgium/Holland/USA/Italy/Germany

Tandata users add SIX zeros (000000) or wait for 3 seconds of modem tone if dialling manually

01344-890987 (up to V34 28800 bps)

See our home page on the internet:

http://www.firshman.demon.co.uk

COMPUTER CLEANERS (UK only)

All in standard 3-pin devices and are plug-in NO WIRING REQUIRED. In their eleventh year of production - as old as the Sinclair QL

2-way adaptor..£14 **3-way adaptor.. £18**
4-way trailing socket.... £24

APR 97

Prices include postage and packing (Airmail where applicable). Prices are: UK (EC/Europe outside EC/Rest of world). Payment by cheque drawn on bank with UK address, debit card/Mastercard/Access/Eurocard/postal order or CASH! (No Eurocheques). Send SAE or 1RC for full list and details

VISA

Holly Corner, Priory Road, Chavey Down, ASCOT, Berks, SL5 8RL
Tel: 01344-890986 Fax/BBS: 01344-890987
tony@firshman.demon.co.uk http://www.firshman.demon.co.uk

MasterCard

anything else as well.

Back to the QMAC review. QMAC does what it claims to do, it assembles source code and does it very quickly indeed. DJToolkit (remember that ?) is around 100K of source code and it assembles in about 8 or 9 seconds on a QL fitted with a Gold Card. On a QXL it is even faster. I don't have a Super Gold Card to test it on, but I am sure it will be quicker still.

One important feature of QMAC is the ability to let you split your source code over a number of smaller and, in theory, easily maintainable source files. References can be made from one source file to another using the XDEF and XREF directives although it can get a bit tedious when you assemble and get an error as you have forgotten to XDEF and XREF all the references that you have used but you get used to it.

One thing that QMAC requires that the previous assembler I used did not, is a SECTION for all code. Sections just group together lumps of code and data and I suspect that I don't really use them to their best as I usually have 3 sections, CODE for all the code, DATA for all the data and, if required, a CONFIG section if I am setting up a standard config block in a job.

COMMON sections are also catered for by QMAC although, as I mentioned above, I don't use them. This is not through choice. I have read the manual and it is not very clear on the needs for sections or common areas - I am sure that Jochen or Phil Borman have a much better idea than I have and might be persuaded to do 'an idiot's guide' to sections & commons.

QMAC can also be used to assemble position dependent - ie fixed - code. On the QL this is not really acceptable these days as almost all QL code is relocatable. This is a different 'relocatable' from the one in SROFF. It means that the code must always be run from a specific start address and not just EXEC'd or CALLed from wherever it happens to be loaded. ROM code for example. Having said that, I suspect that even ROM code these days will be position independent just in case it is loaded into a QXL or similar emulator using the ROM_LOAD command.

QLINK the Linker

The linker part of this development kit is called QLINK. It simply takes the SROFF format file produced by QMAC and 'resolves any unsatisfied fixups' to create a finished code file. All that this means is that any calls to a library routine, or XREF & XDEF etc, references are correctly coded so that when the finished program makes a call to the library or to a routine in another source file, the code gets to the correct place.

QLINK, as with QMAC can be invoked in a number of ways and with a number of command line options. Some of these can be set up as defaults using CONFIG and this saves typing on the command line. The options for executing the linker are the same as described above for QMAC and will not require repeating here.

The command line is limited on both of these programs. I have not worked out how many characters can be typed, but for the linker this is a much more serious problem. To carry out a sucessful link, you need to tell it all the SROFF files, library files and listing options and files that are required. You cannot do a partial link followed by another as the first one will throw out a few 'unresolved reference' errors.

QLINK allows you to use a control file to specify a list of all the options that you need to carry out the link. This file is a simple text file and contains only 3 commands - INPUT, EXTRACT and LIBRARY.

The INPUT command simply tells the linker to include the whole file in the final output. If the filename is that of a library, then every routine in the library will be included in the final output file. This may not be suitable so the EXTRACT command is used.

The EXTRACT command tells the linker to look in the given file for a module name. If it finds it, extract it from the library and include it in the output file.

Of course you might have built up a large library of modules and may not remember all the names. The LIBRARY command lets the library be scanned (once) for any outstanding unresolved references. This could be a problem as the linker will only look once in the library. If it finds a module that itself calls another module that the linker has already passed by in the library then that reference will be unresolved at the end of the link.

To try to avoid this you can specify another LIBRARY command which has the same library name as the first. AT the end of the link, you will have a code file and if requested a load map file and a debug file for use with a symbolic debugger. AT least that's what the manual says. I don't yet know of any symbolic debuggers for the QL.

Libraries

I mentioned that the linker can extr modules from a library or can scan a library any unresolved references. What exactly is library?

A library, according to the manual, is an SRC file which contains one or more modules. module is simply a self contained piece of cc – a subroutine for example, although a mod may make calls to other modules.

When a small source file containing a sin subroutine is assembled by QMAC, you ge file holding a single module in SROFF format library is simply a lot of these module fi appended to one another. There may be so library manager programs available that will this for you.I I seem to remember one in Quanta library many years ago. I suspect it be on the LA01 disc, but check with the lat libguide.

There is a library manager called SLB on C68 discs.

The Macro Library

The macro library supplied with QMAC is qu a good one. There are all sorts of good hidden away in here. I won't mention them all by adding the line

```
INCLUDE macro_lib
```

to your source file, a whole new world assembler coding opens up for you.

The most useful of all the macros is STRING$ macro. It is used as follows :

```
filename STRING$ 'Win1_macro_lib'
```

and when assembled, produces the follow code :

```
filename DC.W 14
         DC.B 'Win1_macro_lib'
```

so you never ever have to count up length of QDOS format strings again. And dc forget what a pain debugging can be when y continually get a 'Not found' error when try to open a file even though you know the file there and that you have spelt the na correctly in your source code – but did y count up the number of characters correctly have you maybe changed the name since y first wrote it an forgot to recount ?

You can also use the STRING$ macro to up any QDOS format string, for example, for names in an EXECable file, filenames, strings be printed on the screen etc etc. This has to the most useful macro in the whole library, a yet, when you look at the source for it – ...

MACRO_LIB - it is only 6 lines long, 4 if you don't include the macro name and the end macro directive. The only problem with it is the fact that if your string has any spaces or commas etc in it, the whole thisng must be surrounded with curly brackets, as per the following example :

```
a_string STRING$ {'This has spaces in it'}
```

Other macros allow you to code assembler source in a sort of high level manner. There are all sorts of structured programming techniques available within macro_lib. For example there is :

IF - ENDIF with THEN ELSE and ELSEIF FOR - ENDFOR with TO and DOWNTO to control the direction and STEP to control the step size. WHILE - ENDWHILE, REPEAT - UNTIL or REPEAT - FOREVER. SWITCH - ENDSWITCH with CASE and ENDC & DEFAULT to control multiple choice jumps etc.

This is very useful, if not a little strange at first. Once you have used these constructs for a while, you might begin to wonder what the fuss about assembler programming is all about !

The manuals

The manuals are a very important part of any software these days. The manual for QMAC covers the assembler, the linker, the macros and the QED editor that is supplied. It does not attempt to teach you how to program in assembler but there are a number of good books on the subject available and there are tutorials in the Quanta library.

The manuals are quite terse in places and at least 2 readings are required when first getting to grips with the package. Some of the information is quite useful - the full list of assembler instructions and the brief explanations of the various addressing modes do help.

The layout of an SROFF file is detailed as is the format of the QMAC and QLINK listing files, symbol tables and map file.

There are a few example source files on the disc and the use of the structured macro facilities is well documented, both in the source code and in the manual although it takes a bit of getting used to, the fact that you can look at an example of how it should be done in conjunction with the manual telling you all about it is very useful.

Summary

Well worth the money if you are going to make any use of assembler on the QL and its derivatives. It is quick in operation and so far has never given me any problems. See the latest Quanta manual for details of who to get it from and how much to pay for it. You won't regret it. When I started to use it, I actually gave away my old assemblers. One was the QMAC predecessor from GST (the non macro version which came on microdrives) and the Metacomco assembler development kit which I paid a much more that £15.00 for back in the old days.

On the disc there are example source files, linker control files, a window manager to set the size of the windows the two programs use - these can be CONFIG'd as well. There is a copy of the CONFIG program and a pair of include files containing the entire set of QDOS traps, vectors, system & SuperBasic variables, channel & window definitions and so on. ■

Cleanup Program

Ian Pizer

When I obtain text via a bulletin board, World Wide Web with LYNX, or by other computer method there are often (usually) many unwanted code sequences like [*H, [*m (where * represents a string of any length), control codes, etc., which makes reading difficult or impossible. To clean up the text pass the text through the following program and find the cleaned text in RAM2_X:

```
100 REMark file to cleanup a text file
110 CLS:INPUT 'name of file to treat › ';f$
120 OPEN#3,f$:OPEN_NEW#4,ram2_x
130 REPeat loopA
140   x$=INKEY$(#3):IF EOF(#3):EXIT loopA
142   IF CODE(x$)=13:PRINT#4:NEXT :REMark CR code becomes real CR
143   IF CODE(x$)=32:PRINT#4,' '; :NEXT :REMark space code becomes real space
145   IF CODE(x$)‹32 OR CODE(x$)›191:NEXT:REMark ditch these codes
```

QUANTA

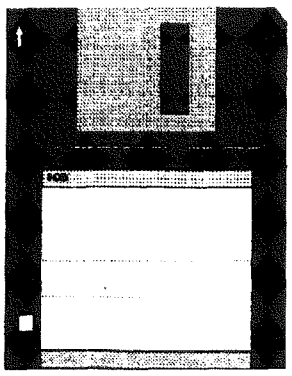
Independent QL Users Group

Worldwide Membership is by subscription
only, and offers the following benefits:
Monthly Newsletter - up to 40 pages
Massive Software Library - All Free !
Free Helpline and Workshops
Regional Sub-Groups. One near you?
Advice on Software and Hardware problems
Subscription just £14 for UK members
Overseas subscription £17

Barclaycard: Visa: Access: Mastercard

Now in our FOURTEENTH successful year

Further details from the Membership Secretary



Bill Newell
213 Manor Road
Benfleet
Essex
SS7 4JD
Tel. (01268) 754407

```
150   IF x$='[' :REMark some junk begins here
160     REPeat loopB
170     x$=INKEY$(#3)
180     IF x$='H':PRINT#4:NEXT: REMark [....H becomes CR
185     IF x$='m' or x$='K' or x$='C':NEXT:REMark [..m or C or K is eliminated
190     END REPeat loopB
200   ELSE PRINT#4,x$; : REMark this is a good text character
205   END IF
210 END REPeat loopA
220 CLOSE#3,#4
230 BEEP 10000,10
```

The result will not be perfect but it makes o messy text quite readable. This program only deals with [*H or [*m or [*K or [*C but you could easily modify the program so it will remove other nuisance sequence of characters. It is worth looking carefully that this program, as is, or modified, does not throw out lumps of useful text, hence the use of Master Spy (see below). Simple repeated annoyances like =20 can be eliminated with most text editors but cannot easily be replaced by a control character.

(LYNX is a program which gets you onto the Web but without images, you only get text). Maybe soon available for QL.

Master SPY

To compare the "dirty" text file with the "clean" result I used Master SPY (MS) which gives the possibility of having simultaneously two adjacent windows which can be manipulated independently. I learnt how to do this from Norman Dunbar (see QL Today Sept/Oct 1996 p48). So one reads one file into Master Spy, then using F4 you can shrink the window width to half the screen with "Size" and arrows. You can now read in a second file (F3 Read), and shrink it with F4 and "Size" and shift it with "Position" so you can see both files. Now you can operate on either windows using "CTRL/down- or up-arrow" to swap the cursor. The shifting and sizing is a little fiddlybut you can quickly learn how to do it.

■

More Sorting

Stephen Poole

Thank you for the article in QL Today on sorting. Please find enclosed a listing in Superbasic which is much faster than the Pigeon Sort. It must be said that it only sorts integers, the highest of which is represented by 'n'. A version for SMSQ sorts 32766 items in 10 seconds, and Turbo runs it even faster. The variable 'pr' can be set from 0 to 1 to print out the results, but this slows the whole process down. I hope you are satisfied with the result.

I have also written an arborescent *[means 'related to trees'! - Editor]* sort for floats or strings, which is almost as fast as the pigeon-sort.

```
100 REMark TALLY-sort by S. Poole v25 Mar 97
110 CLS : ct = 0 : n = 1000 : pr = 1 : REMark n maximum is 32766
120 DIM x(n),y(n) : d1 = DATE
130 FOR f = 1 TO n
140   j = RND(n) : IF pr : PRINT j!!
150   x(j) = x(j) + 1
160 END FOR f
170 IF pr : PRINT \\
180 FOR f = 0 TO n
190   j = x(f)
200   IF j THEN
210     FOR k = 1 TO j
220       ct = ct + 1 : y(ct) = f
230       IF pr : PRINT f!!
240     END FOR k
250   END IF
260 END FOR f
270 PRINT\\DATE-d1
```

"This little routine arrived at the QL Today office on April 1st."

Letter-Box

Al E. Green writes:

Gentlemen and GODS of the QL:

I have had a Sinclair QL for quite a few years now and it took a long time to change it from the standard 128K version with the temperamental microdrives. But when I did, I went all the way. First was the Miracle Dual port Centronics printer interface, then I really took the plunge, I purchased a GOLD CARD and wouldn't you know it the Super Gold Card was released a week later: oh well!!

Next came two ED disk drives and a Hewlett Packard disk case and a 60 watt power supply. (this case was for 2 full height 3.5 inch drives) This gave me room for 2 half height drives plus a place to install a 3.5 inch IDE Hard Drive with one empty bay. I also replaced the IPC chip with a HERMES co-processor.

I ordered the IDE interface from Frank Davis and picked up a 130 Meg hard drive here in Tampa. With all the cables, wires and plugs all over the desk I realized that I would need to change the case to something better. The IBM Mini Tower seem like the best choice.

I did not like the colour of this case so with a can of semi-flat black paint it now looks more like the QL. With the addition of this case, I would need an interface for an IBM keyboard plus a powered ground plane board just to install it. What I am getting at is that I have spent more money on this machine than I have spent on all the others computers that I own (over 30) and I am not done yet. The OS will be replaced with the MINERVA rom.

What I have installed in the IBM case is one 130 meg hard drive, two 3.5 inch ED drives and one 6X IDE CD ROM DRIVE. This is the reason that I am writing this letter. I need software to access this drive. I would like to start with audio disks then at a latter time maybe we could get programs (some graphics) on CD's.

I have had a SVGA monitor for about a year now. I heard a rumor that one could be used with the QL, if so how much difference would I see between the Sinclair Vision monitor vs SVGA?

Unless you use the Aurora card, there is only one important frequency to check for: the QL needs a monitor which can handle a horizontal frequency of 15.625 KHz. Most SVGA nowadays start scanning at around 30 KHz, so they are not suitable. If you find a NEC Multisync II (not IIa!) or, better, a NEC Multisync 3D (not 3FG etc.) then you'll get a much better display than old QL monitors gave you - the picture tube mask is much better.

Last but not least: By any chance do you know of anyone familiar with the Amstrad line of computers? The PCW-8256, PCW-9512, CPC-128, Infomation needed: How to replace their 3.0 inch disk drive with 3.5" 720K drives (26 pin connector to 34 pin connector). PCC-640 boot disk needed.

Anybody out there to help Al?

F. W. Gregory, Packington, England, writes:

First of all, my sincere congratulations to all concerned with the publication of 'QL Today'. It must mean a lot of hard work for you all but I assure you that your efforts are greatly appreciated. As for the cover disk with the last issue, I cannot praise it enough. It is so useful and easy to use. Please continue to produce cover disks as often as you are able.

Now, allow me to endorse the comments of Messrs. Chappell and Ree concerning the plight of beginners. I am one of them! It is no doubt fair to say that we are only a small fraction of those who use the QL in some form or other and only a person like yourself can judge whether they are or not, and by how much, we of the next generation of QL users are worthy of consideration. I use the expression 'next generation' with tongue-in-cheek because I am now retired. But a few years ago my wife bought me a basic QL for Christmas. I was working then, knew nothing about computers and could devote little time to it. Gradually, I have upgraded it and now have a computer keyboard and Gold Card, floppy disk drive and hard disk drive in a tower thanks to Qubbesoft. And, of course, I have much more time.

I have purchased quite a lot of software, some of it I am sure unwisely because I did not understand exactly what I was buying or really for what purpose. In hindsight, I should have approached someone like yourself and sought expert advice as to what software might be regarded as essential and the best value for money.

However, I am now the proud owner of a hard disk because that is what I believed all the experts have. But being a beginner, I now wonder what is the best way to use it. Indeed, what are the different ways of using it and what are the advantages and disadvantages of each of those different ways. And while I can study some BOOT programs in QL Today and various examples in the QPAC2 manual, how do I choose between them and how do I transfer my software from floppy to hard disk so that they

operate without difficulty, e.g. Perfection, Professional Publisher, and so on. I am not into multitasking so wouldn't it be nice to have a simple list come up on the screen telling me that if I press F1 I will get 'Address Book & Lbel Printer', F2 will give me 'Perfection', F3 - 'Archive', and so on. Well, that is one idea, anyway.

What I am really saying is that there seems to be very little written about the use of hard disks and I am suggesting that you might consider publishing a series of articles on this subject in QL Today. What are the different ways of using hard disks for various levels of user, what are their advantages and disadvantages, when do you need to set up WIN2_, WIN3_, etc in addition to WIN1_, and how best to do it, how do you transfer your software onto hard disk?

To my mind one of the big advantages of the QL is that you have choices about what to do and how to do it which I do not believe you have with the PCs bought from the large retailers. But the effective use of these choices requires knowledge which can be acquired easiest by it being passed in simple terms from those who have it to those of us who do not have it to begin with.

I apologise if this letter seems to you to be a rather rambling affair but I hope it serves to indicate to you my appreciation of QL Today and my general thoughts as a QL novice.

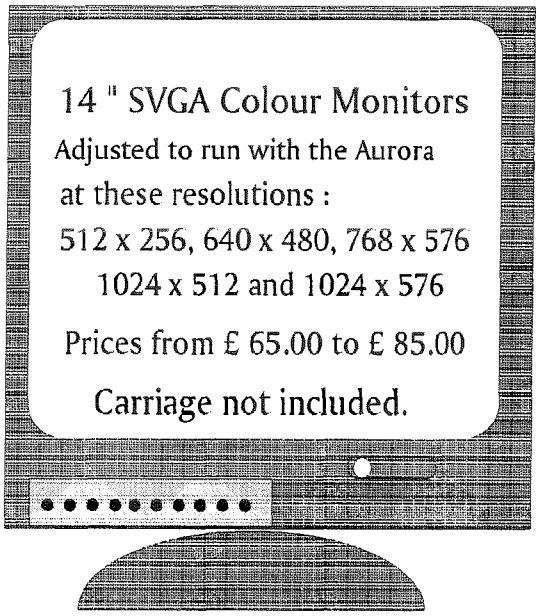
There are several ways, to transfer programs to hard disk, the easiest being to follow any instructions given in the manual for the software in question. The second easiest, and probably most general, way is to make use fo the DEV device, documented in the Gold Card manual. Using this, you will be able to transfer most programs which can run from floppy disk to run from a directory on hard disk by fooling the program into thinking it is running from floppy. First, decide on a name for a directory for a given program, e.g. if using a program called XYZ, you may decide to use a directory called "win1_xyz_". Create this with the command MAKE_DIR "WIN1_xyz_" Now copy the program from floppy to the directory with the command WCOPY FLP1_ TO WIN1_xyz_ Before the program can be run, you need to alter some DEV settings (and you'll have to read the (Super) Gold Card manual for an explanation of DEV). You'll have to do this each time before running the program, so it may be worth saving it a s a little basic program in the directory, or adding it to the program's BOOT file, otherwise it becomes hard to remember and a bit of a chore as well. DEV_USE 1,WIN1_xyz_ DEV_USE 2,WIN1_xyz_ (These two don't have to be the same) DEV_USE FLP FLP_USE FDV

The first two commands tell the system where the disguised floppy drives are to be referenced, so that any references to FLP1_ and FLP2_ in the software actually end up referring to the new directory on the hard disk. Both can be different - for instance, if the program normally runs from FLP1_ and stores its files on FLP2_, like Quill usually does, you could create a second subdirectory called WIN1_xyz_files_ on the hard disk and make the second definition DEV_USE 2,win1_xyz_files The third command changes the name of the DEV device to FLP.This is how the system gets fooled. But this means we have now lost the real floppy drives, as any attempt to DIR FLP1_ will fail because it now lies on the hard disk. But we can get around this by renaming the old floppy system to something else, like FDV, so DIR FDV1_ would now get a list of files on the floppy drive, so we use the FLP_USE FDV command. Most programs will run from hard disk like this, but you are faced with the problem of remembering to reset the system settings afterwards. This is done with: DEV_USE DEV FLP_USE FLP Using DEV in this way, as the manual implies, can be confusing and needs to be well thought out. Quo Vadis Design have a program called DEV Manager which automates much of this after you have initially defined the settings, and for the low price, may be worth looking at.

The main advantages of hard disks are speed and the fact that everything is instantly available off the one drive without the need to exchange floppies all the time. The disadvantages are that drive problems usually mean you lose the whole lot in one go, so you need to regularly make backups to cater for this problem. As Jochen and I know to our costs, it's not IF your hard disk will fail or become corrupted, it's WHEN, so we make backups, even thought the very act of making backups usually means they're never needed, whereas the person who doesn't make backups gets the hard disk crashes (something related to Murphy's law there, I think).

Finally, would anyone like to write a more detailed article on the subject of use of hard disks, to cover the areas set out by Mr. Gregory in his letter? Contact me at the address inside the front cover. - Dilwyn Jones
■

Get a better view



Q Branch
Feeling out on a limb ?
Reach out for Q Branch
Suppliers of Quality QDOS/SMSQ products
Hardware and Software

P.O. Box 7.
Portslade
E. Sussex
BN41 2ND

Tel : 01273-386030
Fax : 01273-381577
Email :
qbranch@qbranch.demon.co.uk

Hardware

QXL II	£ 200.00
Super Gold Card	£ 160.00
Recycled Gold Card	£ 60.00 *
Aurora	£ 120.00
Qubide	£ 55.00
Qplane	£ 25.00
Aurora cables	£ 3.00
Aurora rom adaptor	£ 3.00

* when available.

Package deals

QXL II + SMSQ/E	£ 260.00
Aurora + SMSQ/E	£ 180.00
Aurora + SMSQ/E + Super Gold Card	£ 330.00
Super Gold Card + Monitor	£ 380.00

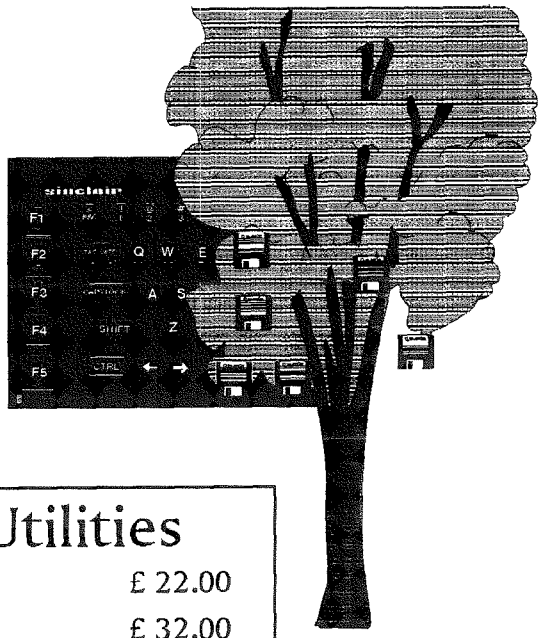
Q Branch News

Super Gold Cards are back - but for a limited time only. By the time you read this we will have made the next batch of Super Gold cards and should have them available for sale. One of the chips is now obsolete and it will be impossible to manufacture any more of these popular memory expansion / disk drive interfaces. Qubbesoft and Miracle do not expect to have their new cards available for some time and we will still be offering generous trade in prices should you wish to trade up later. QCount, the pointer driven accounts package is now released. This program will allow you to keep track of several bank / building society or share accounts and view the contents by using DATAdesign. It comes complete with the pointer environment and the DATAdesign engine. PROGS have released LINEdesign v 2.11 which runs under the ProWesS system and upgrades for existing LINEdesign owners are free. (you do need the latest version of ProWesS for it to run so send both disks + return postage for upgrading). PFdata, the program that prints from the DATAdesign database using ProWesS is now public domain and can be obtained from us. We have a limited amount of SVGA monitors all set up to work with the new Aurora and we are offering package deals for people who wish to put a new system together. Don't forget that you can get our catalogue by sending us a stamped addressed A5 envelope and we have a range of demo disks available too.

Demo disks £ 1.25 + P&P
Demo collections £ 2.00 + P&P
PFdata £ 1.25 + P&P

Q Branch

Feeling out on a limb ?
Reach out for Q Branch
Suppliers of Quality QDOS/SMSQ products
Hardware and Software



P.O. Box 7, Portslade, East Sussex.
BN41 2ND
tel: 01273-386030 fax: 01273-381577
Email :
qbranch@qbranch.demon.co.uk

Q Branch high quality mouse mats
£ 5.00

Programming

QD v 9.04	£ 53.00
QD + Qliberator	£ 69.00
QD + Qliberator + QBasic	£ 110.00
Qliberator	£ 50.00
Master Spy v 3.3	£ 30.00
QPTR	£ 37.00
Easyptr pt 1	£ 37.00
Easyptr ptr 2 & 3	£ 18.00
QMake	£ 18.00
QMon / JMon	£ 22.00
Basic Linker	£ 22.00
DISA 3	£ 37.00
QMenu	£ 16.00

Utilities

FiFi 2	£ 22.00
QSup	£ 32.00
QSpread	£ 66.00
Cueshell 2	£ 37.00
Qload / Qref	£ 15.00
Disk Mate 5	£ 28.00
QPAC 1	£ 40.00
QPAC 2	£ 20.00
QTYP 2	£ 30.00
QLQ	£ 32.00
Ldump	£ 26.00

ProWesS Programs

ProWesS	£ 48.00
DATAdesign	£ 24.00
Fontutils	£ 30.00
File Search	£ 12.00
PFlist	£ 12.00
Fontpack	£ 60.00
LINEdesign v 2.11	£ 24.00

SMSQ/E

Gold Card / Atari / QXL version
£ 76.00
Various Atari versions : call for details

QPC

SMSQ/E owners	£ 78.00
non SMSQ/E owners	£ 100.00

Just Words from Geoff Wicks

Thesaurus, Style Check, Solvitplus 2 £ 15.00 each
two programs £ 25.00 / three programs £ 35.00

Now Available !!

QCount (account software)	£ 25.00	Flashback SE	£ 40.00
Text 87 plus 4	£ 79.00	Flashback	£ 25.00

+ many other programs previously sold by Quo Vadis. - call for details

QTPI

Graham Buck

The choice of Comms software for the Sinclair QL and QDOS/SMSQ computers for those that wish to make use of newer/faster modems is limited some what to one; and that is QTPI by Jonathan Hudson.

How does one get it, and what do you do with it when you have got it are two questions that come to mind, especially if you stop to look at the copious notes and instructions that come with it. Unfortunately, it may well be impossible to answer either question briefly without further adding to any confusion that a new user may have, but the only real answer is that it is a case that you need to enlist the help of someone who already has it, or is prepared to get it for you and then further you need their help in getting it working.

The following is a list of the files that make up QTPI 1.60 as held on the Nene Valley BBS. At the time of writing January 1997 this is the latest version.

QTPI160H.ZIP	27/07/96	54k [0005]	HTML (hypertext) documentation
QTPI160P.ZIP	27/07/96	134k [0003]	PostScript (tm) documentation
QTPI160D.ZIP	27/07/96	62k [0006]	ASCII documentation and script examples
QTPI160E.ZIP	27/07/96	61k [0008]	Fixed cursor keys, ^F, (sh)Hermes detection. New Comm Dev options.
XPRZM352.ZIP	18/05/96	22k [0014]	XPR ZMODEM v3.52 - send file close bug fix
XPR351.ZIP	08/02/96	93k [0031]	XPRs v3.51 for QTPI - Removes screen write timeout - J Hudson

Additionally, to make use of the Scripting language the following files are also required.

CSM122.ZIP	21/09/96	68k [0003]	New integer functions (i.e REQUEST%) and other minor improvements to BASIC interface.
env107.bin	06/04/95	1k [0001]	ENV support v1.07 needed for use with SMSQ v2.47
ENV_zip	23/01/95	11k [0002]	Environmental variable support v1.07. SMS compatible (Dave Walker - QL version)

QTPI also requires the Pointer Environment in order to work, this is the 3 programmes Ptr_gen, Wman and Hot_rext and optionally Menu_rext. If you have SMSQ/E then this includes the Pointer Environment. The QJump Config program or MenuConfig is also required in order to change the default directory in QTPI should you wish to run it from other than Flp1_. TK2 is also required. As all of the above files are in Zip form, you obviously require a copy of UNZip in order to decompress them.

It should be noted that you do not actually need all the files listed above as 3 of them are just the instruction manuals but in different formats to suit the type of printer or software that you have to reproduce them. The working parts of QTPI are on QTPI160E.ZIP and XPR351.ZIP with XPRZM352.ZIP being an update to XPR3511.ZIP but you need both.

Before starting with the software, it may be worth considering the hardware, ideally you require as much expansion as possible Gold Card/Super Gold Card together with Disk Drives or even a Hard Disk. As QTPI is a Pointer Environment program then a Mouse is very handy, but unfortunately if like me you only have a SerMouse, then that won't do, a QIMI or ICE mouse is the preferred rodent. The next problem is the QL serial port which is not all that it should be as it is only reliable up to 9600 baud and all new Modems work much faster than that. There are several solutions or options to this problem, I have opted for a Hermes replacement chip but you could choose either the Super Hermes (with the extra serial and mouse ports etc) or the Terry Harman fast serial board which is the fastest option. If you pick either of the latter options, then the modem connections are solved for you, but for anyone taking the first option, you then need to make up a suitable adaptor (or find someone to make it for you) from the QL serial socket to the usual 9 pin connector found on the modem lead. It is generally accepted that the description in the QL User Guide for the serial port is incorrect and using the instructions from the QTPI manual I was able to make a lead that worked first time:

QL Ser Pin	QL Cable Colour	DB25 pin	DB9 pin	
1 GND	Black	7 GND	5 GND	
2 TXD	White	2 TXD	3 TXD	
3 RXD	Green	3 RXD	2 RXD	
4 DTR	Blue	4 RTS	7 RTS	
5 CTS	Red	5 CTS	8 CTS	
		6 to 20	6 to 4	(connect DSR - DTR)

NOTE: QL serial pin 6 is NOT connected to ANYTHING, this is + 12V and could kill your modem. If you are using a 25 pin socket then pins 6 and 20 need to be linked and similarly with a 9 pin socket pins 6 and 4 need to be linked.

Unzipping the 3 indicated zip files will reveal the following files of which the minimum requirement for a working copy of QTPI is indicated by the asterisks '*' not forgetting of course wman, ptr_gen and hot_rext

QTPI160E.ZIP contains:		XPR351.ZIP contains:		XPRZM352.ZIP contains:	
exe/qtpi	*	xprascii_library	*	xprzmodem_library	*
exe/QTPI_dist	*	xprkermit_library	*		
exe/QBook_dist	*	xprsealink_library	*		
		xprxmodem_library	*		
exe/qtpi-keys_dist		xprymodem_library	*		
exe/QTPIMenu_dist					
		xprzmodem_library			
exe/QTPI160_man		XPR351_txt		XPRZ352_txt	
exe/README					

For my own purposes I made copies of the _dist files and renamed them as _dat and then configured QTPI to look for these _dat files. The main file that then holds all of this information and can be configured with a text editor is QTPI_dat although all of the options can be configured from within QTPI by using all of the various menu options.

Below are copies of the Qtpi_dist file that comes with QTPI and Qtpi_dat as I have configured it to work with a Zoom 14,400 fax modem. Where there is a difference I have shown it with an '*'.

Qtpi_dist	Qtpi_dat	
Emulation=VT100	Emulation=ANSI4	*
Port=ser2	Port=ser2	
Parity=None	Parity=None	
Flow=Handshake	Flow=Handshake	
Translate=Raw Data	Translate=Raw Data	
Baud=19200	Baud=19200	
RX EOL=‹CR/LF›	RX EOL=‹CR/LF›	
TX EOL=‹CR/LF›	TX EOL=‹CR›	*
Del Char=DEL (127)	Del Char=DEL (127)	
VT Wrap=Off	VT Wrap=On	*
XPR Protocol=ZMODEM	XPR Protocol=	*
Beep=Off	Beep=Off	
CSI=Interpret	CSI=Interpret	
Arrow Keys=As Keys	Arrow Keys=As Keys	
7bit keys=No	7bit keys=No	
VT KeyPad=Yes	VT KeyPad=Yes	
Modem=Hayes		
QConnect=Not used		
Job Swap=Shift ^C	Job Swap=Shift ^C	*
Xon/Xoff=No	Xon/Xoff=Yes	*

```
Controls=Interpret
Exit Baud=None
Access=On Line
Scroll=5
Dial Try=4
Dial Wait=60
Cmd Delay=0
Ser Buffer=16384
Start XPR=Info Icon
On Wake=Redraw

Comm Dev=
QC Bits=8 bits
QC Parity=None
QC Stop=1 Stop bit
QC Tx Baud=75
QC Rx Baud=1200
QC Flow=RTS/CTS
QC Line=QMOD Tx/Rx
Modem Init=ATZ\r
Pre Dial=ATD
Post Dial=\r
Aprs Dial=
Disconnect=+++\|\~ATH0\r
Connect=ATH1\r
Answer=ATA\r
AutoAnswer=ATS0=2\r
Reset=+++\|\~ATZ\r
Version=ATI0I1I2I3\r
Get carrier=CONNECT
Drop carrier=NO CARRIER
Phone Book=flp2_QBook_Dat
Send From=ram2_
Receive To=ram2_
UpLoad Dir=ram2_UpLoad_
DownLoad Dir=ram2_DownLoad_
Libr Dir=flp2_
Hot Keys=flp2_QTPI-Keys_dat
Default Log=ram2_QTPI_log
Start with=Log closed
```

```
Controls=Interpret
Exit Baud=9600          *
Access=On Line
Scroll=0                *
Dial Try=0              *
Dial Wait=60
Cmd Delay=7             *
Ser Buffer=16384
Start XPR=Full Screen   *
On Wake=Redraw
CR in log=Yes
Comm Dev=

Modem Init=ATZ\r
Pre Dial=ATDT131,1111111111 *
Post Dial=\r
Aprs Dial=
Disconnect=+++\|\~ATH0\r
Connect=ATH1\r
Answer=ATA\r
AutoAnswer=ATS0=2\r
Reset=+++\|\~ATZ\r
Version=ATI0I1I2I3\r
Get carrier=CONNECT
Drop carrier=NO CARRIER
Phone Book=flp1_QBook_Dat
Send From=ram2_
Receive To=ram2_
UpLoad Dir=ram2_        *
DownLoad Dir=ram2_      *
Libr Dir=flp1_          *
Hot Keys=flp1_QTPI-Keys_dat
Default Log=ram2_       *
Start with=Log closed
Script Log=ram2_Script_log
Start Script=
Stop Script=
User Menu=
```

Rather than deal with the differences in the order they appear; because some of them are just personal choices, I will start with some of the ones that actually make a difference to the working of QTPI. The first time that I tried to get QTPI working with a Zoom modem I found that while everything appeared to work OK when issuing AT commands from the keyboard, the same commands would not work from the Menus' or from the Telephone Directory and if the modem did not lock up, then only half of the number or command would appear on screen. I did manage to find a solution to this problem by changing the AT commands, but this was not the correct solution which I found after about my sixth read of the QTPI manual.

```
Cmd Delay=0                    Cmd Delay=7
```

It states in the manual that if your modem has failed on dial with older versions, then you should start with "Cmd Delay=25" and work down to a minimum safe value. If you do this dialing your own number, it costs you nothing.

I worked down to Cmd Delay=7 which seems to work OK for me although I make use of Mercury and so part of this delay is taken up with handling the extra 13 numbers of my PIN code. It may work with a shorter delay, try it and see.

```
TX EOL=‹CR/LF›                    TX EOL=‹CR›
Job Swap=Shift ^C                 Job Swap=Shift ^C
```

The first time I tried to log on to a PC based BBS I ran into two separate problems which resulted in being locked up and unable to drop the line. Whilst trying to use a message editor I found that the BBS would not respond to the QL key presses, and the exit keys from the editor where ^C which of course is the usual Job Swap keys. The only way out was to turn the modem off and pull out the phone plug for good measure. This is not a popular solution with some BBS's and may get you some adverse comments.

The solution is that if you suspect the BBS may be PC based then set TX=‹CR› without the LF It is possible just to do this in the individual entries in the Phone Book which may be a better solution as I have just discovered that if you replay log files in local mode then in order to get this to scroll down the page it needs to be set to TX=‹CR/LF&et Job Swap=Shift ^C , this means hold down Shift, Control and the letter C at the same time.

```
Pre Dial=ATD                      Pre Dial=ATDT131,1111111111
```

According to the instructions, it should be possible to put a Mercury or other service providers access code (PIN number) in Aprs Dial= , but I could not get this to work and so my solution is to add it on to Pre Dial.

```
XPR Protocol=ZMODEM               XPR Protocol=
```

If you only use one protocol then set it here, else leave it blank and set the protocols in the individual Phone Book entries.

```
UpLoad Dir=ram2_UpLoad_           UpLoad Dir=ram2_
```

I change this because the "Upload_" part was appended to the files that I sent and caused problems at the receiving end, now they just get the file name.

```
Emulation=VT100                   Emulation=ANSI4
```

ANSI4 is the latest/newest terminal emulation that QTPI can handle and so I start with that and if that fails, work backwards.

With the flies renamed and edited as necessary and saved to a disk together with the Pointer Environment and a suitable boot to LRESPR it, it should be possible to Exec / Ex or Exec_W / Ew Flp1_Qtpi and it will load up and work. Previous versions some times used to fall over if some of the files were missing or wrongly named.

to be continued . . . . .

■

No Files on QPAC2

Stephen A. Hall

QPAC2 has been with us quite a while now and those who have negotiated the learning curve of how to implement the facilities it contains would not be without it. The presentation of the on-screen buttons produced from QPAC2's code is largely under the control of the user and consequently a range of customised button frames can be produced. One group of buttons, however cannot be easily modified. These are of course the buttons for QPAC2's internal programs i.e. Files, Exec, Sysdef etc. I found that a string of buttons to call the files program, configured for a range of devices, was very wasteful of screen space and made the button frame unnecessarily verbose. I was able to rationalise these buttons and a few people asked me to reveal how this was done, so here goes.

For example

Files Win1_ Files Win2_ Files Flp1_ Files Flp2_ Files Ram1_ Files Ram2_ would be better proportioned if displayed as: **F Win1_ F Win2_ F Flp1_ F Flp2_ F Ram1_ F Ram2_**

But 'F' doesnt look to HOT does it ?

I looked at the code of QPAC2 and decided that a mere margin fiddler like myself could convince QPAC2 to display a shorter name for 'Files'.

By the way if you do not know how to set up a Files device button, I do it like this

```
1070 BT_SLEEP 'Files';'\dWIN1_'
```

What I needed was a single character that would represent 'Files' which I could somehow get QPAC2 to use instead of the word 'Files'. I worked out how to replace 'Files' with a single disk icon.

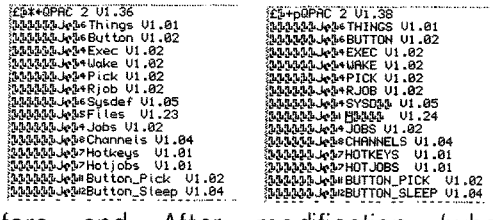
This is what I did.

First, I modified a single character in a copy of the standard QL character font (available from several sources, I can supply this if you don't have it) to look a bit like a disk (see below). I chose character 193, since I have no programs that display this one. The character was originally an arrow pointing up and right, this is easily modified in any of the font editors provided with most design programs i.e. PD3, Qdesign and are available in PD libraries.

Then I modified QPAC2 to use the character 193 instead of 'Files'. This should be done on an uncommitted copy of QPAC2 and requires a good text editor program, I used Master Spy.

The value of the characters in line 9 of QPAC2 are as follows, from left to right 0,0,0,0,0,0, 74,251,0,5 then the word 'Files' (70,105,108,101, 115), then 2 spaces (32,32) then 'V1.23' (86,49, 46,50,51). So the whole line reads: 0,0,0,0,0,0,74, 251,0,5,70,105,108,101,115,32,32,86,49,46,50,51.

Only the brave need read on: I know from past experience that changing the length of executable files can be disastrous, so decided to overwrite only while modifying [thinks! put editor in overwrite mode]. Changing the above line to 0,0,0,0,0,0,74,251,0,1, 193, 0,0,0,0,32,32,86,49,50,51 changes the tenth number from 5 to 1; this is the number of characters in the following name, i.e. 5 for 'Files' to 1 for character 193. 70 to 115 is replaced with 193 and four zeros; replacing Files with our new name (Chr 193) and four zeros to pack out the space.



Before and After modification (whoops! different versions)

If you save this modified version of QPAC2, when you LRESPR it, character 193 will start the Files program.

To force the operating system to use the modified font as the primary font, I use the following lines in my boot file. NOTE: I think that you can't do this with a Sinclair rom only.

Under Minerva:

```
140 sx_base = PEEK_L(VER$(-2)+124)
150 old_font1 = PEEK_L(sx_base+40)
160 font_addr1=RESPR(2306)
170 LBYTES WIN1_FONT_NEWFNT_fnt,font_addr1
180 POKE_L sx_base+40,font_addr1
```

Note: 2306 bytes reserved for a 256 character font.

Under SMSQ/E:

```
380 font_addr1 = RESPR(2306)
390 LBYTES WIN1_FONT_NEWFNT_fnt,font_addr1
400 CHAR_DEF 0,font_addr1
```

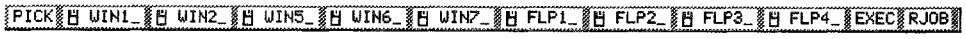
Reference to Minerva or SMSQ/E manuals might help.

I added something like the following lines to my boot file to set up some device buttons:

```
1070 BT_SLEEP  'F';'\dWIN1_'
1080 BT_SLEEP  'F';'\dWIN2_'
1090 BT_SLEEP  'F';'\dWIN3_'
1100 BT_SLEEP  'F';'\dRAM1_'
1110 BT_SLEEP  'F';'\dRAM2_'
1120 BT_SLEEP  'F';'\dFLP1_'
1130 BT_SLEEP  'F';'\dFLP2_'
1140 BT_SLEEP  'F';'\dFLP3_'
1150 BT_SLEEP  'F';'\dFLP4_'
```

Having saved this I used an editor to change the 'Fs'(Chr70) in my boot file before the semicolons to Chr 193.

Thats it; now when I boot up (and the modified QPAC2 is loaded) the Files buttons now present a disk icon and the device name, as you can see.



Worth the effort? I think so, but I would like a better icon design. Any offers?

You will probably notice that I have made some other changes to the start of the QPAC2 file, this is because I prefer capital letters in the buttons. I also shortened SYSDEF a bit (don't forget to change the reference to the button in your boot file if you do this).

One way to get a big Ed

Now while we are on the subject of text editors (we were werent we?), there are to my knowledge three excellent text editors for the QL and compatibles.

QD (ok Jochen)

Master Spy and

DPs' Editor

Each is programme is endowed with special attributes that makes having all of them very useful. But one of them can't use the extended

screen dimensions that Aurora provides: DPs' Editor, right! ... Wrong, it is the configuration programme that won't let you use bigger windows than 512 x 256. Editor works happily at almost 1024 x 576 on my machine (would go bigger) and the large text display area is most welcome.

Right! so how was this amazing discovery made?

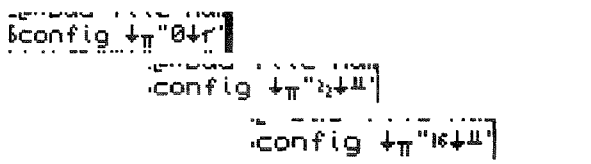
I had been suspicious that Editor might be able to use bigger than standard windows, and during a recent text manipulation session, I desperately needed the command programmability of Editor. Fed up with working in the postage-size window in the corner of the screen I set about finding out where the configuration programme places config'd information within the Editor file. This was easily done by making two identical copies of Editor, then running the configure programme (EDT_CONFIG_EXE) and reconfiguring the screen size (only) in one of the copies. So, I had two copies of Editor, but one had slightly different config info. Naming both copies as source files within DPs' COMPARE (file comparing programme), quickly showed the location of the changed screen size information.

Now to get a different screen size, beyond that which the configuration programme will allow; all you have to do is fiddle the values at the now known location until you get the screen size you require. I loaded a copy of Editor into Master Spy and went to the first known address location (14628 in my copy, but don't rely on it for yours). Addresses 14628 to 14630 (my copy again) held the three bytes that changed, so changing these will redefine the screen size that Editor starts up with. In the 'as supplied' version of Editor the values at these addresses were 48,227,244. I tried changing these and then running the modified version. After a few failed attempts, I found values that would work and provide varying screen sizes. The first and last values, if reduced, increase the window size when the programme is run! Overdoing it simply causes Editor to not start.

These are the values that work for me. (No, I haven't a clue how)

Address	As supplied	1024 x 512	1024 x 576
14628	48	22	16
14629	227	227	227
14630	244	160	160

You can of course reduce the screen size once Editor is running, but don't try and increase it again, it won't work. The version of Editor that I use is 2.05, so any address references may be specific to that version. The location of the config information within the file is easy to find, since it starts with 'config' (See below).



Config Chars as supplied for 1024 x 512 and 1024 x 576 displays

I will try the technique used here to winkle out window size information on other programmes which won't naturally use larger displays, but the proper solution to this problem is to get the config programme altered by the author, so that bigger windows can be specified there (Anyone Know where Chas Dillon is ?).

■

Floating point hardware support for Qdos and SMS

Simon N. Goodwin

This article explains how Qdos and compatible systems have been extended with software which allows any task to take advantage of add-on arithmetic hardware, making mathematical operations much faster. The software is free and speeds up programs written in C68, SuperBASIC, SBASIC, Supercharge, Turbo, QLiberator and GWASS, so far.

It's a significant breakthrough because it makes floating point operations almost as fast as integer ones on a modern 32 bit Qdos emulator. It's been successfully tested on QXLs and Amigas with 68020, 68030, 68040 and 68060 with appropriate FPUs. The software gives compatibility between Motorola's original add-on floating point chips, the 68881 and 68882, and later fully-integrated processors like the 68040 and 68060.

FLOATING WHAT?

Floating point operations involve both digits, or 'mantissa', and a decimal point and an 'exponent' to indicate very large or very small values, such as 1E10 (one followed by ten zeroes) or 2E-2 (a fiftieth). Such values are generally quite useful, but they are much more difficult for a computer than integer operations, which work

on whole numbers in a fairly limited range. But floating point is the default format for BASIC and most other programming languages, and commonly used in graphics, spreadsheets and database applications.

Integers are faster but not as general, so floating point values are often used to be 'on the safe side', at the expense of speed and storage space. When the QL was designed, floating point hardware was rare on microcomputers, expensive and not very fast.

FLOATING POINT UNITS

Since then floating point processors have got cheaper and faster, to the point where they're now an integrated part of modern micro chips - you get one whether you want it or not. Many QXL owners have upgraded to a faster 68040 chip, and found that the replacement part contains floating point (FPU) and memory management (MMU) hardware as a matter of course.

Qdos and SMS emulators have been built around other Motorola-based micro systems, which include floating point hardware or sockets for the extra co-processor chip. This is true of 32 bit Amigas (1200, 3000 and 4000) as well as later Atari machines like the TT and Falcon. So many Qdos users already have floating point hardware, or somewhere to fit the chip, and today 68882 co-processors are cheap.

The original QL 68008 and Gold Card 68000 processors do not implement the interface which allows a co-processor to intercept and handle certain instructions, like the floating point extras. It is possible to add an FPU to those systems, but difficult. Special hardware and software is needed. When the 68020 arrived Motorola built-in support for up to eight co-processors, although it's rare to find more than one FPU, in practice.

Floating point units add instructions to the 68000 machine code repertoire. These include FSQRT, which finds square roots, FADD, FSUB, FDIV and FMUL, which implement conventional mathematical operations. Eight extra registers store temporary results inside the floating point unit, and values can be read and stored with any of the usual 68000 or extended 68020 memory addressing modes.

The floating point instruction set depends on the version of the co-processor. The 68881 and 68882 are equivalent but the later can work almost twice as fast as the 68881, and supports higher clock speeds. It's not essential to match the clock speed of the FPU and the main processor - many of my tests used a cheap 20 MHz FPU with a 25 MHz 68030 - but if the FPU is much faster than the main processor the speed advantage may be squandered.

MICROCODE

6888x floating point units use 'microcode' to break each operation down into a sequence of simple internal actions - typically 50 steps for simple operations like addition and subtraction and over 100 for division and square roots. This sounds slow, but it is still tens of times faster than performing the same operation piecemeal in integer registers, and the main processor can get on with work to which it is better suited while the FPU grinds away.

Microcode let Motorola implement functions like logarithms and trigonometry as well as the more fundamental mathematical operations. The functions used longer sequences of microcode and built-in tables of mathematical constants.

68040 and 68060 floating point units do not use microcode. They include custom hardware to perform the basic mathematical operations, which makes them much faster on the additions and multiplications which are most common in 'real' (sic) programs. These FPUs can generate a new floating-point result every three cycles - ten or twenty times faster than the microcoded FPUs, and almost as fast as integer operations.

The drawback is that the complicated instructions like FSIN and FLOG are not directly recognised. If the 68040 or 68060 finds one of these codes it invokes a machine code support program called the FPSP which works out the desired result using simpler operations, and re-starts the program as if the 'exception' had never been signalled. This mechanism is a bit long-winded - it would be quicker to use the simpler operations directly - but it allows old programs written for a 68881 or 68882 to run at comparable speed on a newer processor, without the need to re-write them.

This mechanism is not used in the current Qdos floating point extensions, as we have no need to support existing programs written with older FPUs in mind. Instead we have implemented a 'library' of code which Qdos tasks can call to perform floating point operations. The loader installs the correct library for your FPU, masking the differences.

The 68040 and 68060 are available in versions with and without floating point hardware. The 'EC' and 'LC' versions lack floating point units, and cannot be connected to external

68881 or 68882 FPUs, so the only way to make such systems support floating point hardware is to replace the chip with a full-blown 'MC' or 'XC' version. This is no problem on existing 040 processor boards like the QXL, normally supplied with an EC040, or Commodore's 3640 (sometimes fitted with an LC040) as the chip pin-out is the same, so all you have to do is unplug one chip and install another.

This is not difficult but it requires care as the chips are valuable, fragile and nailed down by about 200 gold-plated pins. If you order a replacement chip for one of these boards, make sure you get the PGA (Pin Grid Array) version, as the 68040 and 68060 are also available in cheaper PLCC (Plastic Leaded Chip Carrier) packages which are intended for direct soldering onto a board, and won't fit a PGA socket.

THE PIONEER

The first QL compatible system to use floating point hardware was CST's Thor 21, a rare but remarkable machine. This was based on a Sinclair QL circuit board, with a 32 bit 68020 processor and 68881 floating point unit piggy-backed on. This was an unhappy combination because the eight bit QL memory ran at a fraction of the speed expected by the new processors, which squandered most of their speed twiddling their thumbs waiting for values to crawl between processors and memory.

CST's David Oliver rewrote Qdos to use the extra instructions of the 68020 and 68881 where they could yield faster results. The problem is that Qdos uses a six byte format for floating point values, whereas Motorola's hardware uses the four and eight byte formats standardised by the American Institute of Electrical and Electronic Engineers, known as IEEE single and double precision.

The QL format is a compromise, ideal for a processor with 32 bit registers, more precise than the seven digit IEEE single precision format but rather restrictive compared with IEEE double precision, which can represent values with 16 significant digits.

In fact Motorola co-processors use a TWELVE byte 'extended IEEE' format internally, and all values are converted to that form before they're used in calculations. David worked out a fairly quick way to convert values between Qdos six byte format and the double-sized Motorola extended format, and replaced the SuperBASIC functions that make heavy use of floating point operations with code to convert values to and

from IEEE format and do all the hard work with FPU instructions.

This approach is great for functions like EXP and ATAN, but marginal for simpler operations like multiplication and division, when the overhead of format conversion swamps the time saved working out the result. The only solution to this would be to change BASIC to use the IEEE format consistently throughout, and while this might be a good idea in the long run, it's a lot of work for both SMS SBASIC and Qdos SuperBASIC, and sure to cause incompatibility as the format of values will change.

The Thor 21 took an intermediate route, keeping values in Qdos format for multiplication and division, but using 64 bit 68020 instructions to speed them up a bit. I've got patches which do this for Amiga Qdos, but they're of marginal benefit and not yet ready for release, even though they should work on any system with BASIC in RAM and a 68020, 68030 or 68040 processor. Contact me by EMAIL if you'd like to patch these into your system, and think you know basically how to do it.

IMPLEMENTATION

To make full use of your FPU, you need some PD files. First you must LRESPR FPSAVE_BIN, which checks the type of FPU fitted, recording the details in system variables at $D0(A6) then looks in the current directories (unless re-configured) for an FPSP for your FPU. If all goes well FPSAVE loads the FPSP and all SMS or Qdos tasks can share the FPU without unwanted interactions. At this stage the only known problems are with Atari systems - we're investigating these, with help from Jochen Merz. We'd like to hear from other Atari+FPU owners.

If you want your S(uper)BASIC programs to take advantage of the FPU, you also need to load my FPUFNs. These are FPU-based versions of the standard BASIC transcendental functions. Normally you'd load FPUFNs_OVER_ CODE which (on "JS" Qdos or later) overwrites the old functions with new FPU ones. After this all your BASIC programs, whether interpreted or compiled, will use the FPU without any modification.

For convenience when testing speed and accuracy, FPUFNS_TEST_CODE contains the same code but with all the function names prefixed with the letter F so you can compare the results of, say, FSQRT with SQRT, or use the test program FPU_TIMES_BAS to compare the speed. This uses DIY Toolkit TIMING_CODE extensions for accurate timing.

EXAMPLE PROGRAM

FPSAVE, FPUFNs and the various FPSPs are much too long to list here, and freely available on disk or by modem. I have written a small program which lets you try out your FPU with no need for other files. This adds one function, FSQRT, which is like Sinclair's SQRT function but uses the FPU. It's not as efficient as the FPUFNs version, because it cannot rely on FPSAVE to prevent unwanted task interaction, but it works, letting you test your FPU easily and see some of the potential benefit.

You don't even need to type in the assembly code, as the DIY Toolkit Hex loader returns (hurrah!) to automatically load the code from hexadecimal DATA statements. This example is short by DIY standards, so you only need to type in about ten lines of DATA to get it working, if you've already entered the hex loader for a toolkit project.

```
100 REMark Ex-Sinclair QL World HEX LOADER v 3
110 REMark by Marcus Jeffery & Simon N Goodwin
120 :
150 CLS: RESTORE : READ space: start=RESPR(space)
160 PRINT "Loading Hex..." : HEX_LOAD start
170 INPUT "Save to file...";f$
180 SBYTES f$,start,byte : STOP
190 :
200 DEFine FuNction DECIMAL(x)
210 RETurn CODE(h$(x))-48-7*(h$(x)>"9")
220 END DEFine DECIMAL
230 :
240 DEFine PROCedure HEX_LOAD(start)
290 byte = 0 : checksum = 0
300 REPeat load_hex_digits
310    READ h$
320    IF h$="*" : EXIT load_hex_digits
330    IF LEN(h$) MOD 2
340      PRINT"Odd number of hex digits in: ";h$
350      STOP
360    END IF
370    FOR b = 1 TO LEN(h$) STEP 2
380      hb = DECIMAL(b) : lb = DECIMAL(b+1)
390      IF hb<0 OR hb>15 OR lb<0 OR lb>15
400        PRINT"Illegal hex digit in: ";h$: STOP
420      END IF
430      POKE start+byte,16*hb+lb
440      checksum = checksum + 16*hb + lb
450      byte = byte + 1
460    END FOR b
470 END REPeat load_hex_digits
480 READ check
490 IF check <> checksum
500   PRINT"Checksum incorrect. Recheck data.":STOP
520 END IF
530 PRINT"Checksum correct, data entered at: ";start
560 END DEFine HEX_LOAD
570 :
580 REMark Space requirements for the machine code
590 DATA 144
600 :
610 REMark Machine code data
```

```
620 DATA "43FA007E34780110","4ED2347801144E92"
630 DATA "666C534366663031","E800674C2231E802"
640 DATA "6B56064037FE4E40","42A7D2812F0142A7"
650 DATA "3E80F2174800F200","0004F21768003017"
660 DATA "672C222F0004E289","640852816404E291"
670 DATA "5240044037FE6B16","3380E8002381E802"
680 DATA "4FEF000C027CDFFF","780270004E754271"
690 DATA "E80042B1E80260E8","70EE4E7570F14E75"
700 DATA "000000000001FF84","0546535152540000"
710 DATA "*",12821
```

The assembler listing shows exactly how values are converted from Qdos to IEEE format, and back again. This is the main work done by my FPUFNs, with some optimisations and checks for tricky values. The difficult mathematics is all handled by the FPSP routines, optimised for a particular FPU.

In the case of SQRT there's no risk of overflow and all inputs and outputs are positive, which simplifies the conversion but requires an extra check to reject negative values. In the absence of FPSAVE, the actual FPU work is all done in Supervisor mode (after TRAP #0) so no other task can use the FPU register at that critical time. This is inefficient, but it's simple and it works.

FSQRT is written in assembly code, and suits any QL assembler as the new FP instructions are written as constants. GWASS is generally best for FPU development as it recognises all the new instruction mnemonics. FSQRT is directly supported by all Motorola FPUs unlike, say, FSIN and FEXP which require the FPSP on a 68040 or 68060.

```
* Replacement SuperBASIC square root function by FPU maths
* Simon N Goodwin, Amiga Qdos 3.24 November 96..April 1997
*   Assemble with GWASS for FPU opcode support, or use the
*   DC.Ls. Code suitable for 68040/68060/68881/68882 FPUs.

bias        equ         14334          FP exponent difference

start       lea.l       define,a1
            movea.w     ($110).w,a2
            jmp         (a2)           Add function to BASIC

fsqrt       movea.w     ($114).w,a2    Get FP parameter
            jsr         (a2)           Use Qdos vector
            bne.s       fail
            subq.w      #1,d3          One at a time
            bne.s       oops

* Conovert via 96 bit IEEE format, like the Thor 21, to
* ensure adequate exponent range and minimise shifting.

            move.w      0(a1,a6.l),d0
            beq.s       done           SQRT(0)=0, easy!
            move.l      2(a1,a6.l),d1
            bmi.s       range          Must be positive
            addi.w      #bias,d0       Bias exponent
            trap        #0             Prevent context swaps
            clr.l       -(a7)          IEEE.X value low bits
            add.l       d1,d1          Normal 31 bit mantissa
            move.l      d1,-(a7)
            clr.l       -(a7)          Last of 3 long words
            move.w      d0,(a7)        Plug in exponent

* Perform the SQRT operation in IEEE floating point format
*
*             fmove.x     (a7),fp0     Load value into FPU
*             fsqrt       fp0
*             fmove.x     fp0,(a7)

              dc.l        $F2174800      Above FPU opcodes
              dc.l        $F2000004       expressed for old
              dc.l        $F2176800        Qdos assemblers

* Convert positive result from IEEE to Qdos format

            move.w      (a7),d0        Exponent
            beq.s       zero
```

```
            move.l      4(a7),d1        Mantissa
            lsr.l       #1,d1           Qdos positive value

* Perform rounding according to the 32nd (discarded) bit

            bcc.s       rounded
            addq.l      #1,d1           Round
            bcc.s       rounded
            roxr.l      #1,d1           Divide mantissa by 2
            addq.w      #1,d0           Double exponent

rounded     subi.w      #bias,d0        Adjust exponent
            bmi.s       zero            Minimal

stack       move.w      d0,0(a1,a6.l)   Stack exponent
            move.l      d1,2(a1,a6.l)   Stack mantissa

tidy        lea.l       12(a7),a7       Deallocate extended
            andi.w      #$DFFF,sr       Leave Supervisor mode

done        moveq       #2,d4           Signal Qdos FP result
            moveq       #0,d0
            rts

zero        clr.w       0(a1,a6.l)      Return FP 0
            clr.l       2(a1,a6.l)
            bra.s       tidy

range       moveq       #-18,d0         Negative root?!
            rts

oops        moveq       #-15,d0         Bad parameter signal
fail        rts

define      dc.w        0,0             No procedures
            dc.w        1               One function
            dc.w        fsqrt-*
            dc.b        5,'FSQRT'       Or SQRT to upgrade
            dc.w        0

            end
```

The code should be easy to follow if you know the format of Qdos and IEEE extended precision numbers. It checks the sign of the input, sifting out the 'trivial' case of zero, and shifting it left one place (with ADD.L) so it is 'normalised' in IEEE terms. Qdos keeps the sign in the most significant bit of the mantissa, unlike IEEE. The result is shifted down and rounded correctly.

The constant BIAS is used to convert Qdos exponents to and from the corresponding IEEE values, which allow a wider range. The temporary IEEE values are stored on the Supervisor stack, six of which are zero after unpacking the six byte Qdos format into 12 byte IEEE eXtended form.

FPU FUNCTIONS

If you want BASIC programs to use the FPU the easy approach is to replace the slowest floating point functions with eponymous ones that use the FPU internally. This does not speed up operators like multiplication and division, but it does boost the functions that make most use of software floating point normally. You don't get the full benefit of the FPU, because of the overhead of format conversion and checking, but the technique stays compatible with existing programs - results should be identical, but faster - and the same extensions suit original SuperBASIC and the new SBASIC.

The full FPSP is needed for my FPUFNs_CODE, a set of replacements for all the BASIC transcendental functions. These have no need for the TRAP #0, so they're even faster. At present they use 20 digit IEEE extended precision internally, but this seems gratuitous when Qdos only needs half that, so I'd like to hear from any users who reckon they can optimise Motorola's code to trade between speed and precision. If you're mathematically minded, can read assembly language, and think you can help, please contact me by Email:

simon@studio.woden.com

FPSAVE automatically detects which tasks are using the FPU and does not need to intervene for the majority, which do not use it. But perhaps this will change, as others join George, Dave Walker and me in extending Qdos to make full use of the FPU.

The FPSP is freely available, in versions for 68881 and 68882 FPUs with either 68020 or 68030 main processors (only the FPU is relevant) and much larger versions for 68040 and 68060. The '040 version is biggest, at almost 50K, but it does include quite a lot of routines that are not generally available in BASIC, such as hyperbolic functions and optimised logarithms for various bases. The FPSP interface for programmers is the same regardless of FPU - you just put the operands on the A7 stack and call a subroutine from a table at the start of the FPSP which leaves the result in FP0. The latest versions also include functions to convert IEEE values to and from Qdos FP format, similar to those I optimised for C68.

You also need George Gwilt's FPSAVE utility to allow tasks to share the FPU safely. You'll need GWASS to re-assemble any of the FPU code, which is written in 68020 assembly language. All these programs are available, with source, for free, but you may need to buy an FPU.

Make sure you get a set of files which includes the VER extension, to check the FPSP versions, as earlier test versions have a few bugs - unfortunately it's difficult to clear these old versions out of the PD chain, unless users fobbed off with old programs complain! FPSAVE should be version 1.17 or later.

TEST TIMINGS

The table gives test timings for the prototype FPUFNs on several machines: an Amiga 4000/030 with 68882, Thierry Godefroy's 32 MHz QXL with XC68040, George Gwilt's standard speed 20 MHz QXL (also with a full 68040) and my 50 MHz A4000/060. Speed without FPU is 100%.

	030/25	040/32	040/20	060/50
SIN	1142 %	307 %	544 %	300 %
COS	3138 %	391 %	575 %	318 %
EXP	1105 %	1180 %	658 %	372 %
SQRT	1200 %	400 %	–	1905 %
TAN	1030 %	792 %	1053 %	351 %
COT	944 %	751 %	914 %	454 %
ATN	1065 %	439 %	569 %	272 %
ASIN	1574 %	472 %	503 %	549 %
ACOS	1497 %	460 %	500 %	427 %
ACOT	–	–	–	258 %
LN	748 %	329 %	468 %	295 %
LOG10	880 %	323 %	419 %	278 %

The first columns shows the time for a 25 MHz 68030 with a 20 MHz FPU - a 68882. There are no results for a 68020+68881 combination yet. It should come close to the 68030 figure, depending on relative clock speeds.

The boost is most with a separate FPU, but that's misleading as the integer unit is relatively slow, giving the FPU an advantage. The 68040 and 68060 FPUs work best when given a steady stream of code to process, rather than just the odd instruction in a sea of integer operations. For this reason C68 shows the most impressive performance boost on 68040 and 68060 processors.

Results depend on the range of values used. ACOT was a late arrival. Thierry's test used much bigger values than expected, which crashed COT. I've since added a range test to reject values that SuperBASIC would not permit anyway, and the checks on EXP,ASIN and ACOS seem watertight. SQRT took so little time on George's 20 MHz QXL that it gave a meaningless negative 'speed-up' factor over 20,000 iterations, so we've implemented a null function (FNULL) in the test suite which makes SMS timings more accurate.

SBASIC is a lot faster than SuperBASIC, which makes the loop overhead much less significant. SuperBASIC users often compile their code to avoid delays caused by Sinclair's slow interpreter, and if you do that the FPU functions give proportionally much more benefit. Even so, they only speed up the dozen functions listed, and while they're among the slowest on the QL - though still VERY fast compared with other 'eight bit' micros - you won't see any benefit unless your programs use those functions!

CREDITS

This project was the brainchild of George Gwilt, an early Thor 21 owner who went on to write GWASS, the only Qdos assembler that supports the extra instructions of later 680XX processors, including hardware floating point and memory management. This was subsequently used by Dave Walker to add FPU support to C68, and by me to augment BASIC.

Having completed GWASS, George went on to develop FPSAVE, the program that identifies the FPU and makes it appear that every Qdos or SMS task has its own co-processor, and did considerable work on Motorola's FPSP. George Gwilt is the main architect of Qdos and SMS floating point support, and it would not have happened without him. CST's David Oliver also deserves credit for working out the basics of FPU support for Qdos almost ten years ago.

I'm grateful to the pan-European test team of Thierry Godefroy, Jochen Merz, Davide Santachiara, Mark J Swift, John Tanner and Dave Walker, who checked out FPSAVE and the FPUFNs on a wide range of hardware. I was amazed how many people have already got FPU hardware, even before it was supported by Qdos, and I'm sure that number will increase now it benefits existing programs.

■

The QL Show Agenda

Saturday, 24th of May 1997	The Netherlands, Eindhoven, St. Joris College - same venue as always.
Sunday, 15th of June 1997	Germany, Solms, Taunushalle. See below for details.
Saturday, 19th of July 1997	London QL Workshop and show. 10:00 – 16:00. St Helen's Church Hall, St Quintin Avenue, LONDON W10. Free parking. Nearest tube: Ladbroke Grove. Organiser: Tony Firshman (see TF Services ad for contact nos).
Saturday, 6th of Sept. 1997	The Netherlands, Eindhoven, St. Joris College - same venue as always.
Sunday, 5th of October 1997	Byfleet, Surrey, England. Quanta Workshop. More details follow in the next issue.

2nd Sunday every Month	Quanta London Sub-Group meeting. St. Aloysius School Hall, Phoenix Road, NW1. Meeting starts at 13:00 and ends at 18:00. Contact Basil Lee 0181-789-1976 or basilee@mail.bogo.co.uk

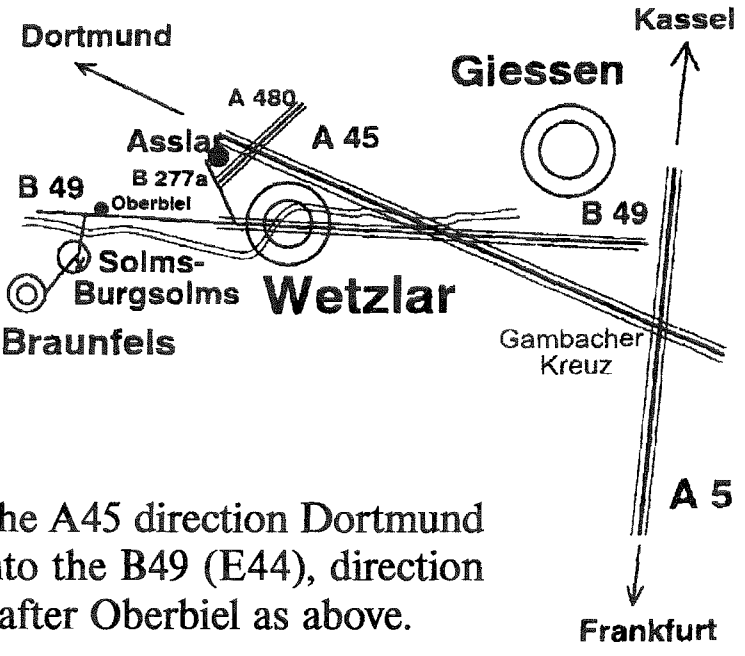
If you need more details, please contact your local QL user group or QL Today!
You can help to turn QL shows into even more interesting events!

Also, if you (plan to) organise QL shows or regular local user group meetings, please let us know and we publish it here!

The German QL Show

15th of June, Solms, Taunushalle

If you come from the West (A45 from Dortmund), leave the A45 at cross Wetzlar and move via A480 and B277a to the B49 (E44) direction Weilburg/Limburg. You cross the river "Lahn" 4km after "Oberbiel" when you head for Solms-Burgsolms. When you are in Solms follow the main road direction Braun fels and you will see the Taunushalle lefthand.



If you come from the North go onto the A45 direction Dortmund and leave exit Wetzlar Ost directly onto the B49 (E44), direction Weilburg/Limburg. Pass Wetzlar and after Oberbiel as above. From south or East as above.